CURSO PRÁTICO 26 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC – PROGRAMAÇÃO DE JOGOS – CÓDIGO DE MÁQUINA

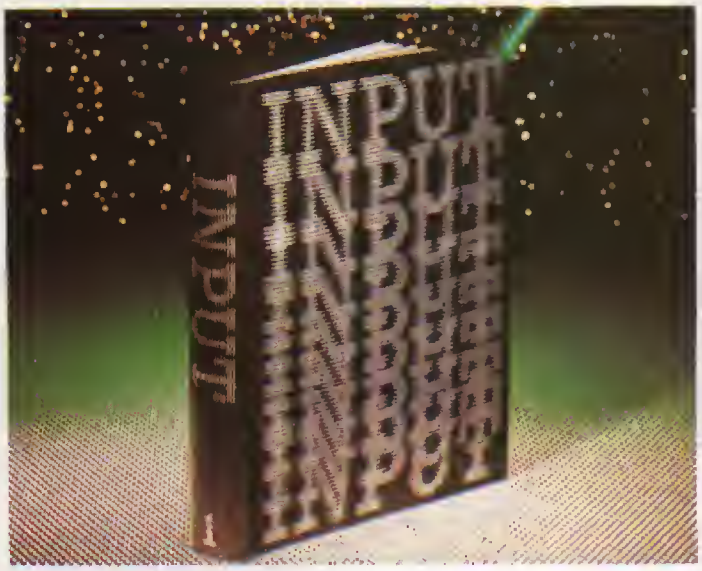
NOVA CULTURAL

Cz$ 39,00

INPUT

Vol. 2 Nº 26

NESTE NÚMERO



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo*, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André. No *Rio de Janeiro*, av. Mem de Sá, 191/193, Centro.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues,
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sânia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Secretária de Redação/Coordenadora: Stefania Crema
Secretário de Redação: Mauro de Queiroz

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas-SP)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Marcos Huascar Velasco,
Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

CLC
A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez,
Menahen M. Politi, Renê C. X. Santos,
Stélio Alves Campos

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda.
e impressa pela Companhia Lithographica Ypiranga.

APERFEIÇOE SUAS TELAS

- A IMPORTÂNCIA DE PLANEJAR
- COMO POSICIONAR O TEXTO
- MELHORE A DIAGRAMAÇÃO
- ADICIONE COR

Para dar a seus programas uma aparência profissional, você precisará planejar cuidadosamente a página-título e outras telas. Aqui você aprende a fazer isso.

Do ponto de vista do usuário, há várias diferenças importantes entre um programa simples e bem planejado e outro que não apresenta essas qualidades. A primeira coisa que distingue um programa "profissional" de um amadorístico é, sem dúvida, o funcionamento adequado, sem erros e sem rotinas inúteis ou mal aproveitadas. Além disso, num nível mais teórico, o programa propriamente dito deve ser bem estruturado e claro. As técnicas necessárias para lhe assegurar uma boa estruturação e a ausência de erros já foram vistas em artigos anteriores.

Porém, mesmo o programa mais bem-feito parecerá obra de um principiante se não tiver uma boa apresentação e, sobretudo, se as telas não forem bem organizadas e fáceis de compreender. Para isso, cada tela, bem como o menu ou mensagem que contiver, deve ser cuidadosamente estudada. Sem o posicionamento correto das declarações PRINT e INPUT será impossível criar telas de apresentação clara e interessante.

No artigo da página 146, examinamos os vários comandos BASIC disponíveis no seu computador que servem para controlar a posição dos caracteres na tela. Agora, você verá como usá-los para montar uma página-título para um jogo imaginário — "INPUT". Técnicas similares poderão ser empregadas para a elaboração de telas que apresentam instruções, menus e mensagens.

INSTRUÇÕES CLARAS

Nada causa pior impressão do que uma tela contendo mensagens com erros de grafia ou sintaxe.

Você já deve ter observado em muitos jogos mensagens do tipo "Você tem um canhão disponível". Este caso particular desabona o programador, sobretudo porque o erro poderia ter sido facilmente corrigido com uma declaração **IF...THEN** (IF L=1 THEN PRINT "CANHÃO DISPONÍVEL"). Antes de começar a se preocupar com os aspectos visuais da tela, procure assegurar-se de ter eliminado todos os erros desse tipo de programa.

Ao diagramar uma tela, leve em conta algumas regras básicas. Em primeiro lugar, não perca de vista o objetivo principal: a clareza. Palavras muito próximas umas das outras dificultam a leitura — assim, deixe um espaço razoável entre as linhas. Evite também dividir palavras; se for impossível, utilize, então, o hífen para separar as duas partes.

Caso precise imprimir na tela dados fornecidos pelo usuário — como um nome, em uma lista de recordes —, tenha o cuidado de evitar que o novo dado interfira nos demais. No exemplo citado, você poderia incluir uma rotina que não aceitasse nomes com mais de um certo número de caracteres ou que, independentemente do tamanho do novo dado, o colocasse numa posição que não afetasse a do placar ou de outras informações exibidas.

Sempre que você tiver uma lista de informações, faça com que todos os dados comecem numa mesma coluna. A recomendação parece óbvia, mas vários programadores não procedem assim.

O segundo aspecto a observar diz respeito à quantidade de informações. Se você incorrer no erro de introduzir informações em excesso numa mesma tela, quem utilizar seu programa não conseguirá memorizar todas as instruções e, em conseqüência, não explorará todas as suas possibilidades.

Por outro lado, para não ser obrigado a trabalhar com um número exagerado de telas, você não poderá reduzir muito a quantidade de informação em cada uma delas. O ponto de equilíbrio dependerá, naturalmente, do seu programa — quanto maior sua facilidade de uso e clareza, menos instruções serão necessárias.

O USO DE CORES

Torne a tela o mais interessante possível. Se puder use cores. Elas dão mais vida à tela, facilitando a compreensão das mensagens ou ressaltando determi-

nados itens. Estude cada caso para decidir se utilizará cores no fundo, no texto, em ambos ou simplesmente na composição de uma moldura.

Os usuários do Apple ou do Sinclair Spectrum dispõem de um comando muito versátil — o **FLASH**. Ele apresentará sua mensagem piscando na tela (em cores, no Spectrum). Impedindo que as instruções assumam uma forma muito estática, esse recurso torna o texto bastante atraente.

Se você possui outro modelo de micro, poderá obter um efeito similar ao do comando **FLASH**, imprimindo o texto repetidamente no mesmo ponto da tela, em cores diferentes.

O comando **INVERSE**, do Apple, também possibilita o destaque de mensagens. Ele é especialmente útil, já que não se pode usar cor na tela de texto desse computador.

POSICIONAMENTO

Para assegurar a clareza da tela, falta ainda um importante elemento: o posicionamento das palavras. Para ter um exemplo, digite e execute este pequeno programa. Você verá *como não fazer* uma página-título para o seu programa. As palavras estão jogadas na tela: não há coordenação entre uma e outra, o que resulta numa grande confusão visual. Mais tarde, examinaremos como arrumar tudo isso.

```
10 PRINT "Apresentamos um nov
o jogo        chamado input"
20 PRINT AT 5,17;"(c) 1986"
30 PRINT AT 12,13;" por Nova
Cultural"
40 PRINT AT 18,18;"qualquer t
ecla"
50 PAUSE 100
60 CLS : STOP
```

```
10 CLS 4
20 PRINT @40,"INPUT"
30 PRINT @136,"copyright nova c
ultural";
40 PRINT @228,"APRESENTA"
50 FOR T=1 TO 2000:NEXT:CLS
```

Para o TRS-80, use apenas **CLS** na linha 10 e multiplique por 2 todos os valores das instruções **PRINT@**.

```
10 PRINTTAB(4);"novacultural ap
resenta"
20 PRINTTAB(10);"input"
30 PRINT:PRINT"copyright 1986"
40 FORJ=1TO500:NEXT
50 CLS
```

```
10  PRINT  TAB( 5);"NOVACULTURA
L APRESENTA"
20  PRINT  TAB( 22);"INPUT"
30  PRINT : PRINT "COPYRIGHT 19
86"
40  FOR J = 1 TO 1000: NEXT
50  HOME
```

Observe o resultado: trata-se, sem dúvida, de uma tela inexpressiva, em que vários detalhes precisam ser alterados. Inicialmente, poderíamos empregar letras maiúsculas para melhorar o efeito. A versão para o Spectrum utiliza letras minúsculas até mesmo para a primeira palavra da tela e para o nome do jogo. Maiúsculas ressaltam palavras importantes ou, se forem usadas no texto inteiro, garantem à tela uma aparência bem clara.

Deveríamos, também, limpar a tela, antes de mostrar um novo conjunto de mensagens. Como você pôde observar no programa anterior, fragmentos do que estava na tela misturam-se à mensagem, completando a confusão.

Não há espaço entre as palavras NOVA e CULTURAL. Deveria haver. Veremos, adiante, como deixar tudo isso em ordem.

Outra falha do programa é não esperar que se pressione alguma tecla para prosseguir. Assim, não há muito tempo para ler as mensagens antes que elas sejam apagadas. Para evitar esse problema é sempre interessante incluir uma linha dizendo o que se deve fazer para que o programa continue — quando o usuário quiser.

No Spectrum, sabendo-se o tamanho da tela e o número de caracteres que compõem as palavras, pode-se planejar com muita facilidade o posicionamento da mensagem.

Suponhamos que seu computador tenha 22 linhas e 32 colunas; você quer colocar uma frase de dez caracteres na segunda linha, e bem no meio dela. Ao verificar o tamanho da frase, não se esqueça dos espaços.

Para calcular a coordenada horizontal da posição onde a frase começará a ser impressa, subtraia o tamanho da frase (10 no nosso caso) do total de caracteres por linha (32) e divida o resultado por 2.

No nosso exemplo, a resposta é 11. Como o 0 é considerado, o primeiro caractere da linha tem coordenada 0, e não 1. Assim, devemos subtrair 1 do resultado acima para conseguir a coordenada horizontal do **PRINT AT** (ou **PRINT TAB**).

Se você quiser posicionar uma frase a apenas dois ou três caracteres da margem esquerda, será mais fácil colocar o número de espaços dentro da declaração **PRINT** em vez de usar **PRINT TAB** ou **PRINT AT**. No entanto, se o número de espaços for grande, evite essa técnica, pois ela tende a gastar muita memória.

Você pode calcular a posição vertical como calcula a horizontal — ou seja, para colocar a mensagem no centro da tela, vê quantas linhas ela ocupa, subtrai esse número do total de linhas e divide por 2. Por fim, subtrai 1 do resultado, considerando que a coordenada da primeira linha é 0.

Quando quiser colocar a linha em outra posição, use um papel gráfico para visualizar melhor a tela, ou, se já tem alguma experiência, estime a posição que lhe convém, de memória.

Os usuários do Spectrum devem observar que a instrução **PRINT AT** funciona de modo um pouco diferente do dos outros computadores. O primeiro número que acompanha essa instrução refere-se à coordenada vertical (y) e o segundo, à coordenada horizontal (x). No entanto, o comando **PLOT** utiliza primeiro a coordenada x e depois a y.

O número após o **PRINT@** refere-se a uma posição da tela de texto do computador. Existem 512 posições no TRS-Color e 1024 no TRS-80. Os números, assim, não podem exceder 511 e 1023, respectivamente, já que a numeração começa no 0.

O cálculo do número da posição que se quer é muito simples. Primeiro, multiplique o número da linha desejada por 32 (esse é o número de caracteres por linha, no Color) ou por 64 (se você está usando um TRS-80). Depois, adicione um valor de 0 a 31 (no Color) ou de 0 a 63 (no TRS-80), para obter a posição horizontal.

Se preferir, deixe que o computador faça as contas por você. Para isso, utilize a seguinte fórmula: **PRINT@ L*32+C, "MENSAGEM"**. L representa a linha que você deseja (de 0 a 14) e C a coluna (de 0 a 31). No TRS-80, use

64 em lugar de 32; C pode variar de 0 a 63.

Para centralizar uma frase na tela, subtraia o total de caracteres da frase do total de caracteres da linha (32 ou 64); divida o resultado por 2 e subtraia 1. Você obtém, assim, o número da coluna em que a impressão da frase se iniciará. Esse valor pode ser utilizado, por exemplo, com a fórmula dada anteriormente. Lembre-se de que os números que seguem a instrução **PRINT@** devem ser inteiros.

MSX

O posicionamento de mensagens no MSX também é muito fácil. Esse computador usa o comando **LOCATE**, seguido da coluna e linha (nesta ordem). Assim, para colocar a mensagem no lugar desejado, basta calcular a coluna e a linha. O mesmo comando também pode ser usado com um argumento numérico, que será interpretado como a coluna onde se iniciará a impressão.

Como o MSX tem duas telas de texto, verifique qual delas está em uso para, então, calcular a posição adequada de impressão. A primeira (**SCREEN0**) apresenta quarenta colunas por 24 linhas e a segunda (**SCREEN1**), 32 colunas por 24 linhas.

Para centralizar uma frase em determinada linha, subtraia o total de caracteres da frase do total de caracteres da linha e divida o resultado por 2. Esta será a posição para o início da frase.

O Apple e seus compatíveis empregam, além do **PRINTTAB**, os comandos **HTAB** e **VTAB** para posicionar o cursor dentro da janela de texto. O comando **HTAB** determina a posição horizontal com valores que variam de 1 a 40 e o **VTAB** determina a posição vertical, com valores que vão de 1 a 24. O uso de valores fora dessas faixas provocará o aparecimento de uma mensagem de erro.

Para a centralização de uma mensagem, primeiro calcule o total de caracteres dela, depois subtraia de 40 e divida por 2. O resultado obtido deve ser utilizado como posição inicial de impressão da mensagem.

O interessante, nesse computador, é a facilidade com que se pode manipular a janela de texto. Essa característica torna muito simples a limpeza da linha de uma determinada posição até o fim dela ou de uma posição até o fim da tela. Utiliza-se o comando **CALL -868** para apagar tudo o que estiver à direita da posição do cursor até o fim da linha. O **CALL -958** limpa todos os caracteres da janela de texto, da posição do cursor até o fim da tela. Desse modo, podemos trocar mensagens em locais específicos da tela, conservando o resto intocado, com grande rapidez e versatilidade.

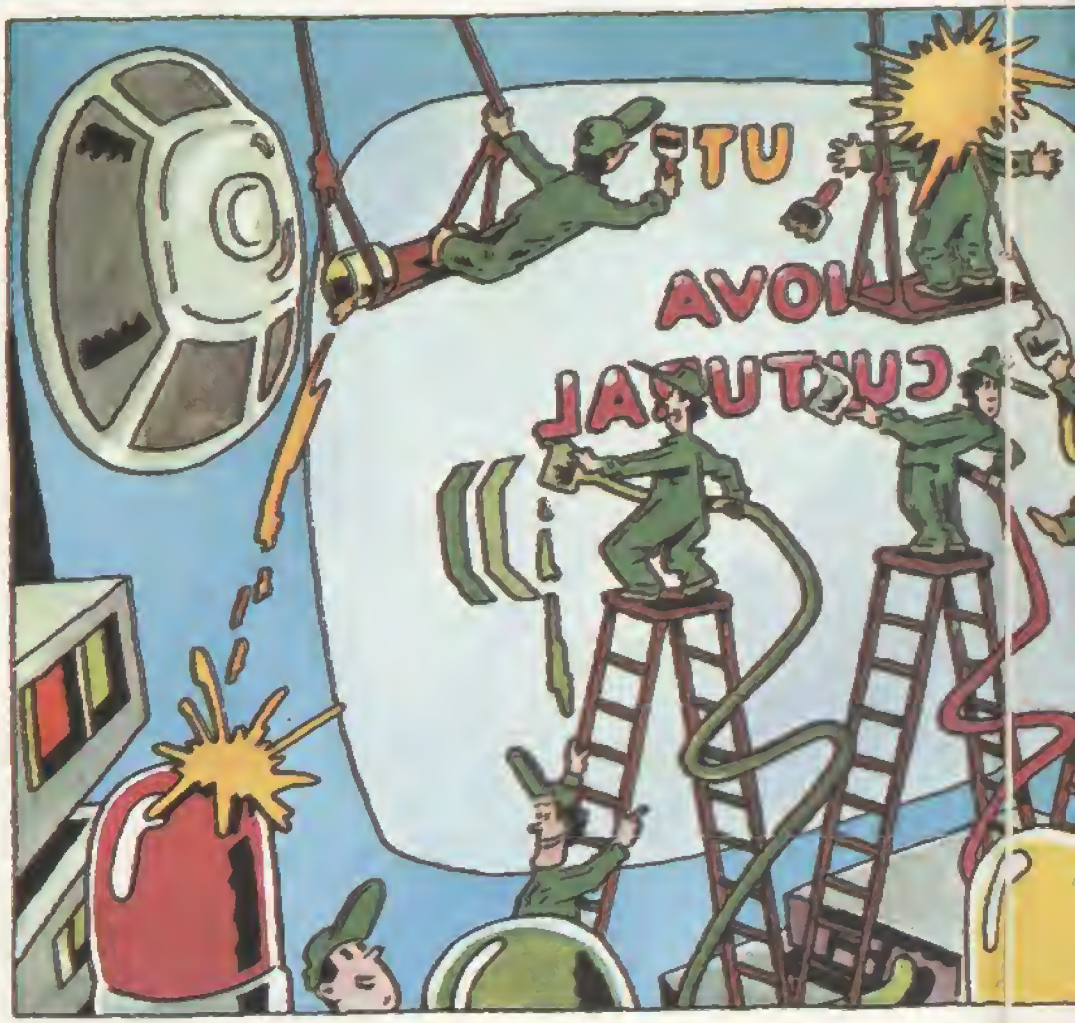
UMA TELA BEM-FEITA

Uma tela pode ser clara e atraente mesmo que não empreguemos técnicas de programação complicadas. O programa dado a seguir é um exemplo disso. Ele corrige os defeitos do programa anterior e adiciona cor e animação ao título, para torná-lo mais interessante. Dê especial atenção ao uso dos comandos de controle do cursor, pois eles são fundamentais em tudo o que diz respeito à impressão na tela.

```
 5 LET X=1
10 BORDER 1: PAPER 1: INK 7:
CLS
20 PRINT  INVERSE 1;AT 3,4;"
EDITORA NOVA CULTURAL "
30 PRINT  INVERSE 1;AT 5,10;"
 APRESENTA "
40 PAUSE 50
50 PRINT  PAPER 6; INK 1;AT
10,10;" I N P U T "
60 PRINT  PAPER 6; INK 2;FLASH
1;AT 9,9;"▝▀▀▀▀▀▀▀▀▀▀▀▀▀▘
";AT11,9;"▗▄▄▄▄▄▄▄▄▄▄▄▄▄▖
70 PRINT  PAPER 6; INK 2;
FLASH 1;AT 10,9;"▌";AT 10,21;
"▐"
80 PRINT  PAPER 5; INK 0;AT
15,2;"COPYRIGHT AUDIO,VISUAL,
1984"
90 PRINT '''" QUALQUER TECLA
 PARA CONTINUAR"
100 PAUSE 0
110 CLS
120 PRINT "INPUT - INDICE"
130 PRINT
140 FOR X=1 TO 10
150 READ a$
155 READ b$
160 PRINT 'TAB 3;a$;TAB 25;b$
170 NEXT X
180 DATA "Animacao","26-32","B
asic,programacao","2-7","BREAK
,Spectrum","7","Cassetes","25"
,"Gravadores","24","CHR$ ,uso
do","26-27","CLEAR","10-27","C
LOAD,Dragon","14","CLS,explica
cao de","27","CODE, Spectrum",
"8"
```

Como você pode observar, o programa, inicialmente, muda a cor da tela e das bordas, e limpa a tela. Escolha as cores de sua preferência, mas procure evitar o habitual fundo negro com letras brancas. Qualquer outra coisa terá efeito melhor, simplesmente por ser diferente.

Não se esqueça, porém, de que a facilidade de leitura é indispensável. Assim, para as letras, opte por uma cor que contraste bem com o fundo. Fora isso, use seu bom gosto: algumas combinações são, sem dúvida, bem mais agradáveis que outras.



POSICIONAMENTO

As linhas 20 e 30 colocam na tela as palavras "EDITORA NOVA CULTURAL APRESENTA". Observe que o nome da empresa fica na primeira linha e a palavra "APRESENTA", duas linhas abaixo. Esse detalhe garante ao texto muito maior clareza.

O programa utiliza o comando **PRINT AT** para posicionar as palavras no centro da tela. Tente descobrir quais deveriam ser os números dessa instrução por meio dos cálculos explicados anteriormente. Veja se eles conferem com os do programa!

Ao contrário do programa anterior, este posiciona as palavras no lugar adequado — detalhe que, como você pode notar, determina uma grande diferença de qualidade entre as telas.

Pode-se também utilizar a instrução **PRINT TAB** para posicionar a mesma palavra, mudando a linha 30 para:

30 PRINT 'TAB 10; INVERSE 1; "APRESENTA"

O apóstrofo junto ao TAB 10 faz com que o Spectrum deixe uma linha em branco antes de imprimir "APRESENTA" na tela. Pode-se também obter o mesmo resultado simplesmente colocando uma instrução **PRINT** vazia (com um número de linha apropriado).

O programa faz uma pausa após essa mensagem para dar maior ênfase ao próximo item que deverá ser impresso, o nome INPUT.

As linhas 50, 60 e 70 imprimem INPUT com uma borda que pisca em vermelho e amarelo. Essa borda é feita com caracteres gráficos da ROM — como se vê no programa — com o comando **FLASH** usando vermelho e amarelo.

Essas linhas utilizam uma série de comandos **PRINT AT** e mostram o quanto ele é flexível. Não há necessidade de se repetir o PRINT enquanto um ponto e vírgula é colocado entre os itens da linha.

Os vários **AT** são separados por ponto e vírgula. Caso os separássemos apenas por vírgulas, estas colocariam meia

linha de espaço na tela, o que poderia apagar alguma outra coisa que ali se encontrasse. Assim, a não ser que você queira apagar o que estiver nesta posição da tela, use de preferência o ponto e vírgula.

Como a borda piscante não fica no centro da linha, o cálculo de suas coordenadas não pode ser feito da maneira explicada anteriormente. Calcule as posições tomando as coordenadas da palavra central da linha e adicione e subtraia o número necessário para obter os resultados para a borda.

Tomemos, como exemplo, a primeira linha da borda. Já sabemos que as coordenadas da palavra INPUT são 10, 10. Sabemos também que esta linha fica uma acima de INPUT. Então, subtraindo 1 do primeiro número, obtemos a primeira coordenada: 9.

Como queremos que a borda comece um espaço antes da palavra, novamente subtraímos 1, agora da segunda coordenada de INPUT, obtendo 9.

Calcule os números das outras partes das bordas e compare seus resultados às coordenadas que empregamos.

O programa completa a tela com uma mensagem de direitos autorais (para lembrar aos usuários que é ilegal copiar programas) e informa que se deve pressionar qualquer tecla para continuar. Ao fazê-lo, não aparecerão instruções nem o início de um jogo, mas um índice de INPUT.

Quando se pressiona uma tecla, o Spectrum limpa a tela e imprime a mensagem "INPUT — INDICE" em seu topo. A mensagem é posicionada no canto esquerdo da linha. Se quiser centralizá-la, introduza nove espaços entre as aspas e a primeira letra da mensagem. Em seguida, o computador deixa uma linha em branco, resultado do **PRINT** sem nenhuma mensagem da linha 130.

O laço **FOR...NEXT** que se segue lê duas variáveis alfanuméricas da linha **DATA** de número 180. Pode parecer estranho que números de páginas estejam armazenados em variáveis desse tipo. Observe, no entanto, que às vezes encontramos mais de um número. Sua armazenagem numa variável numérica seria interpretada como uma subtração e, quando imprimíssemos o valor da variável, iríamos obter o valor –6, o que causaria muita confusão!

Além disso, alguns dados apresentam informações separadas por vírgula. Como você sabe, esta é a pontuação que separa diferentes itens numa linha **DATA**. Assim, é necessário colocar esses dados entre aspas para que o computador não os interprete como coisas distintas — com as aspas, ele lê a informação como um bloco, sem separá-la.

A linha 160 controla a impressão das informações da linha **DATA**, determinando sua posição na tela.

O **PRINT TAB** alinha os dados e números de páginas para que todos comecem na mesma coluna, dando à lista uma aparência organizada. Ele é muito útil na impressão de índices, determinando a coluna em que o dado começará a ser impresso. Nosso programa coloca a indicação na coluna 3 e a página na coluna 25.

Observe que usamos apenas um **PRINT** para imprimir os dois dados na linha, sendo que os **TAB** são separados por ponto e vírgula. Se você mudar o **TAB 25** para **TAB (25 + 32)** ou **TAB 57**, não notará nenhuma diferença. Quando se emprega um número maior que 32, ele é dividido por 32 e o resto é desprezado. O cursor não muda de linha até que se utilize o retrocesso — ou seja, até que um TAB tenha um valor menor do que a posição atual do cursor. O micro pula, então, para a linha seguinte e nela coloca a informação.

Experimente mudar os valores do TAB para ter uma idéia melhor do que eles significam em termos de posição na tela. Lembre-se de que esta tem 32 caracteres por linha.

O **PRINT AT** e o **PRINT TAB**, embora muito parecidos, são usados em circunstâncias diferentes. Sempre que você quiser imprimir uma lista de palavras ou números na tela, utilize **PRINT TAB**. **PRINT AT** é mais poderoso para a impressão de mensagens isoladas em determinado ponto da tela.

T

Digite este programa e veja como o **PRINT@** pode ser usado para produzir uma página-título adequada:

```
10 CLS 3:B$=CHR$(128)
20 PRINT @71,B$;"nova";B$;"cult
ural";B$;
30 PRINT @138,"APRESENTA";
40 PRINT @358,B$;"copyright";B$
;B$;B$;"1986";B$;
50 PRINT @232,CHR$(190);STRING$
(11,CHR$(188));CHR$(189);
60 PRINT @264,CHR$(234);" I N P
 U T ";CHR$(229);
70 PRINT @296,CHR$(155);STRING$
(11,CHR$(147));CHR$(151);
80 PRINT @448," QUALQUER TECLA
PARA CONTINUAR  ";
90 J=1-J:SCREEN 0,J
100 FOR K=1 TO 200:NEXT
110 IF INKEY$<>"" THEN 130
120 GOTO 50
130 CLS 4
140 PRINT @7,"INPUT - INDICE";
150 FOR X=3 TO 12
160 READ D$,E$
170 PRINT @32*X+1,D$;:PRINT @32
*X+25,LEFT$(" "+E$+"      ",7);
180 NEXT X
190 DATA ANIMACAO,26-32,"BASIC,
PROGRAMACAO",2-7,"BREAK,DRAGON"
,7,"CASSETE,FITAS",25,"CASSETE,
GRAVADORES",24,"CHR$,USO DE",26
-27,CLEAR,"10,27","CLOAD,TRS",1
4,"CLS,EXPLICACAO",27,"CODE,SPE
CTRUM",8
200 GOTO 200
```

O programa começa mudando a cor da tela para azul, na linha 10. O resto da linha iguala **B$** a um quadrado escuro. As linhas 20 e 40 imprimem as palavras em caracteres reversos, usando B$ como um espaço. Lembre-se de que os caracteres reversos do TRS-Color aparecem em minúsculo nas listagens.

A linha 30 é similar, com uma diferença: como a palavra APRESENTA é menos importante, aparece em caracteres maiúsculos normais.

No centro da tela está o título INPUT rodeado por um bloco gráfico colorido. Os caracteres e o título são impressos pelas linhas 50 a 70.

O manual indica os caracteres gráficos que estão disponíveis. Eles são constituídos por áreas verdes e áreas escuras. Adicionando-se um múltiplo de 16 ao código do caractere, o verde pode ser substituído por amarelo, azul, vermelho etc.

As áreas da tela coloridas de verde são aquelas em que a cor de fundo aparece. É fácil mudar a cor da tela do verde para o laranja e produzir um efeito pisca-pisca. Para a mudança, a linha 90 usa o comando **SCREEN** de maneira similar à utilizada para mudar a cor de gráficos de alta resolução. **SCREEN 0,1** muda a cor da tela para laranja e **SCREEN 0, 0** traz de volta o verde. O programa troca a cor da tela a cada execução do laço.

Para que se possa apreciar melhor a mudança de cor, a linha 100 estabelece uma pequena pausa. A linha 120 vai fechar o laço.

Se alguma tecla é pressionada enquanto a página está sendo mostrada, a linha 110 faz com que o programa passe para a parte seguinte.

A linha 130 muda a tela para vermelho. A linha 170 e o laço **FOR...NEXT** das linhas 150 e 180 encarregam-se de imprimir os dados lidos da linha 190. A cada execução do laço, uma nova linha é usada para a referência e o número de linha. O **LEFT$** faz com que o número da linha apareça sempre num painel do mesmo tamanho, o que assegura um visual bastante claro.

O toque final cabe ao laço da linha 200 que, embora não pareça, é bem importante. Sem essa linha, o programa colocaria uma mensagem de prontidão na tela, desfigurando nosso trabalho.

Vejamos uma tela mais cuidada para apresentar o nosso jogo:

```
10 COLOR 15,12,12:KEYOFF
20 SCREEN1:CLS:Z$=CHR$(1)
30 PRINTTAB(8);"NOVA CULTURAL"
40 PRINT:PRINTTAB(6);"A P R E S
 E N T A"
50 LOCATE 11,9:PRINTZ$+CHR$(88)
;:FORX=1TO5:PRINT Z$+CHR$(87);:
NEXT:PRINTZ$+CHR$(89)
60 LOCATE 11,10:PRINTZ$+CHR$(86
);"INPUT":Z$+CHR$(86)
70 LOCATE 11,11:PRINTZ$+CHR$(90
);:FORX=1TO5:PRINT Z$+CHR$(87);
:NEXT:PRINTZ$+CHR$(91)
80 LOCATE 5,19:PRINT"Copyright
(C) 1986"
90 LOCATE 2,22:PRINT"Tecle a ba
rra de espaços"
100 IF INKEY$=""THEN100
110 CLS:LOCATE 7:PRINTCHR$(207)
;"INPUT - INDICE";CHR$(208)
120 FOR J=1TO7: LOCATE 0,2*J+6
130 READ A$,B$
140 PRINTA$;TAB(23);B$
150 NEXT
160 GOTO 160
170 DATA ANIMAÇAO,26-32,PROGRAM
AÇAO BASIC,2-7,"CTRL-STOP, MSX"
,8,SISTEMA OPERACIONAL,25,CONTR
OLADORES DE DISCO,24,"CHR$, uso
 do",26-27,"POKE, como usar o",
27-28
```

O programa começa mudando a cor da tela para verde, com caracteres brancos, e desativando as indicações das teclas de função. Em seguida, seleciona a tela de texto de 32 colunas e define a variável **Z$** como o caractere de código 1. Adiante, veremos a utilidade dessa variável.

As linhas 30 e 40 colocam na tela, de forma organizada, a mensagem "NOVA CULTURAL APRESENTA". O **PRINT TAB**, com um valor adequado, encarrega-se de centralizar as palavras, enquanto o **PRINT** vazio que inicia a linha 40 produz o espaço de uma linha entre as palavras.

As linhas 50 e 70 imprimem o título INPUT dentro de uma moldura formada por caracteres gráficos. Esses caracteres são acessados pela instrução **PRINT** na forma: **PRINT CHR$(1)+CHR$** (código do caractere + 64).

Observe o uso da instrução **LOCATE**, já explicada anteriormente, para posicionar na tela caracteres gráficos e palavras.

A linha 100 pode parecer um tanto estranha, mas é muito importante no programa. Ela faz com que o computador varra repetidamente o teclado, até que pressionemos alguma tecla, passando para a parte seguinte.

A linha 110, por sua vez, limpa a tela e posiciona uma mensagem no topo desta. As referências e números de páginas são lidos da linha **DATA** e impressos pelas instruções do laço das linhas 120 a 150.

A linha 160, finalmente, tem como função evitar que o cursor e um OK provocados pelo término da execução do programa atrapalhem o visual da tela.

```
10  HOME
20  INVERSE : PRINT : HTAB 12:
PRINT " NOVA CULTURAL "
30  PRINT : HTAB 10: PRINT " A
P R E S E N T A "
40  VTAB 12: PRINT  SPC( 16): N
ORMAL : PRINT " INPUT ";: INVER
SE : PRINT  SPC( 17)
45  NORMAL : VTAB 20: HTAB 15:
PRINT "VERSAO 1.1"
50  PRINT : HTAB 11: PRINT "COP
YRIGHT (C) 1986"
60  FLASH : VTAB 24: HTAB 7: PR
INT " TECLE A BARRA DE ESPACOS"
;: NORMAL
70  GET A$
80  HOME : INVERSE : PRINT  SPC
( 13);"INPUT - INDICE"; SPC( 13
): NORMAL
90  FOR J = 1 TO 7: VTAB J * 2
+ 8
100  READ I$,P$: PRINT  TAB( 7)
;I$; TAB( 30);P$: NEXT
110  GOTO 110
120  DATA    ANIMACAO,26-32,PRO
GRAMACAO BASIC,2-7,"CTRL-C, App
le",7,SISTEMA OPERACIONAL,25,CO
NTROLADORES DE DISCO,24,"CHR$,
uso do",26-27,"POKE, como usar
o",27-28
```

O programa produz uma tela bem mais elaborada que a anterior. Logo que se inicia, ele limpa a tela e seleciona o modo invertido de apresentação de caracteres. A instrução **PRINT** vazia nas linhas 20 e 30 faz com que o computador pule uma linha antes de começar a imprimir as mensagens.

Os comandos **HTAB** e **VTAB**, como já explicamos, posicionam o cursor para a impressão.

Na linha 40, a instrução **PRINT SPC** (..), no modo invertido, provoca o aparecimento de uma linha escura antes do título INPUT, dando-lhe destaque.

As linhas 45 a 50 imprimem no modo normal mensagens de *copyright* e da versão do programa.

A linha 60 utiliza o comando **FLASH**. Os usuários do TK-2000 devem substituí-lo por **INVERSE**. A mensagem aparecerá piscando no Apple, juntamente com o cursor (é quase impossível escondê-lo, mas, dessa forma, ele se confunde com a mensagem).

A linha 70 faz o computador esperar até que alguma tecla seja pressionada para continuar a execução.

Em seguida, o programa limpa a tela novamente e coloca a mensagem "INPUT - INDICE" no modo invertido. O laço das linhas 90 e 100 lê e imprime na tela as referências e números de páginas respectivos. Observe que isso é feito por meio dos comandos **READ** e **PRINT TAB**. A mesma instrução **PRINT** imprime a referência e o número da página, estando os **TAB** separados por ponto e vírgula. Esse processo facilita muito a montagem de listas e tabelas.

A linha 110 impede que o programa termine. Assim, evita-se o aparecimento do cursor na tela, o que afetaria a distribuição das mensagens.

GERAÇÃO DE BLOCOS GRÁFICOS (2)

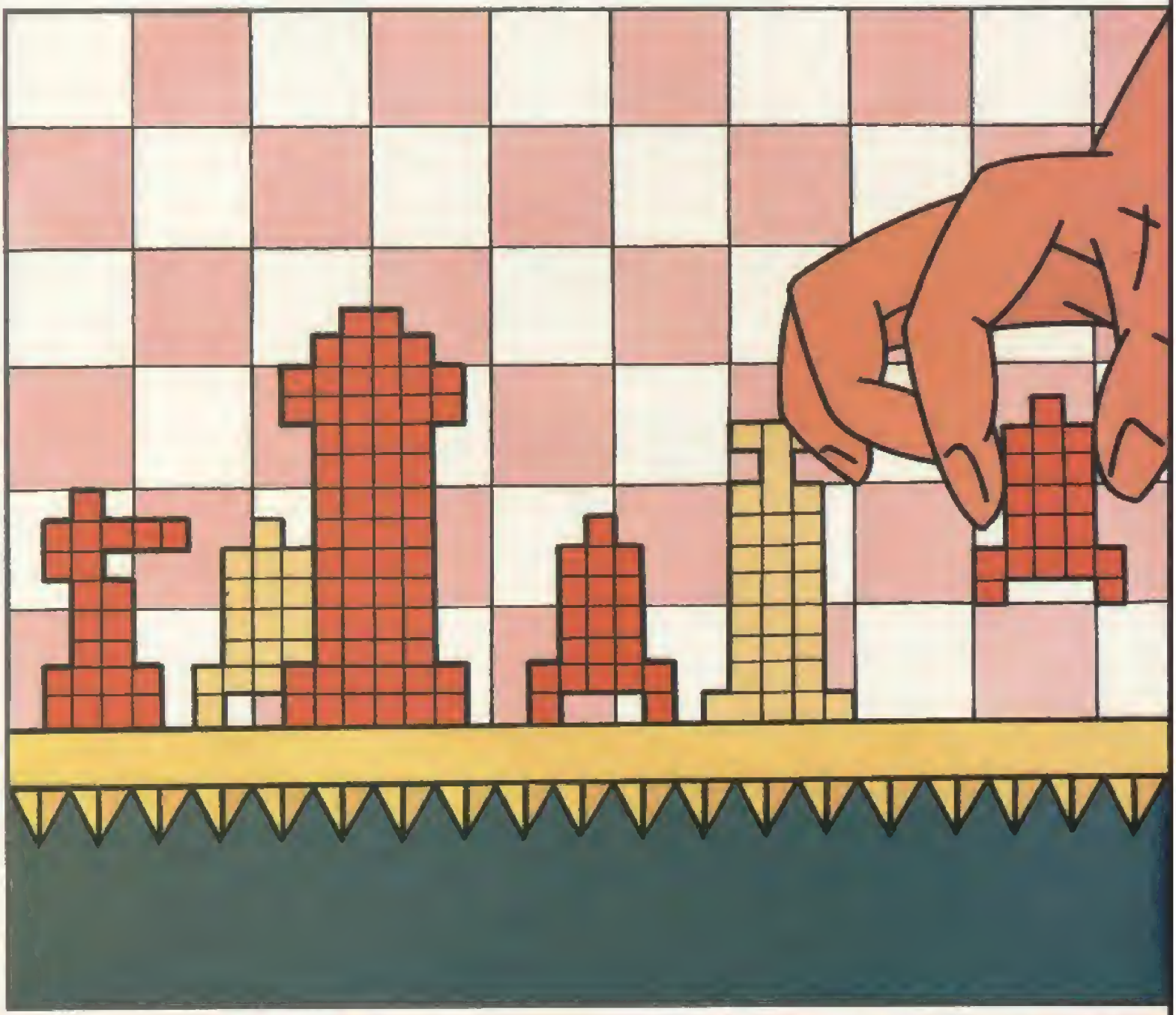
- NOVAS FUNÇÕES
- INVERSÃO DE CORES E FORMAS
- IMAGENS AO ESPELHO
- COMO RODAR OS CARACTERES
- O USO DA IMPRESSORA

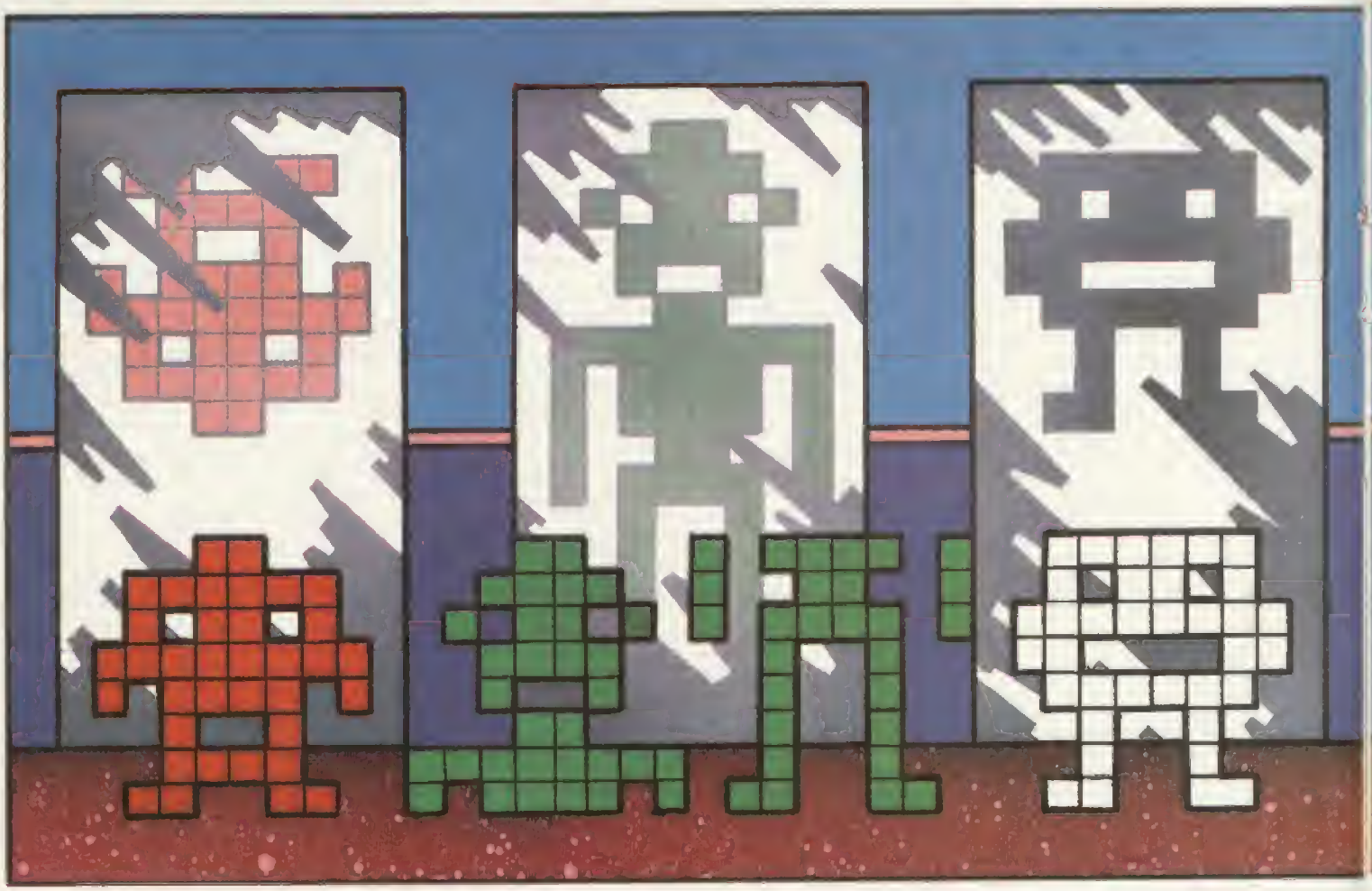
Aqui está a parte que faltava a seu programa gerador de blocos gráficos. Com as novas funções incorporadas, você poderá criar caracteres com muito mais facilidade.

Este artigo completa o programa gerador de caracteres gráficos. As novas linhas acrescentam-lhe recursos que tornam ainda mais fácil a criação de blocos definidos pelo usuário.

Além da parte final do programa, você encontrará aqui algumas dicas de como usar os caracteres criados em seus programas BASIC.

Depois de acrescentar as novas linhas





ao programa dado no artigo anterior, você terá mais seis funções à sua disposição.

A primeira evidencia-se já no início. Ela mostra na tela os valores dos oito bytes correspondentes às linhas do bloco, bem como uma versão em tamanho real na tela. Isso é feito por um programa em linguagem de máquina, que atualiza tanto os valores como o bloco em tamanho real sempre que modificamos um ponto.

O programa completo também permite que você limpe o quadriculado por meio da tecla C, evitando o trabalho de apagar ponto por ponto.

Existem ainda três outras teclas de controle a que se pode recorrer para modificar o padrão do quadriculado. Se você apertar o M, o bloco será substituído por sua imagem ao espelho. Assim, sempre que quiser obter uma figura composta por dois blocos simétricos — uma espaçonave, por exemplo —, basta desenhar um deles, guardá-lo no banco de memória e criar a imagem simétrica usando a tecla M. Também lançamos mão desta função quando estamos desenhando duas figuras iguais, cada qual apontando para uma direção.

De modo semelhante, é possível rodar o bloco 90 graus, pressionando a tecla R. Esta função será muito útil quando se precisar criar apenas um bloco — de um torpedo, digamos — e depois virá-lo em todas as direções.

Com a última das três funções você obterá o inverso do caractere, por meio da tecla I. Se usá-la duas vezes seguidas, terá o inverso do inverso, ou seja, o caractere original.

O programa passará a dispor, também, de uma rotina de impressão. Pode-se empregá-la para imprimir os valores dos bytes que serão colocados em uma linha **DATA**, por exemplo, ou para fazer uma cópia da tela. Esta sairá melhor ou pior de acordo com o tipo de sua impressora.

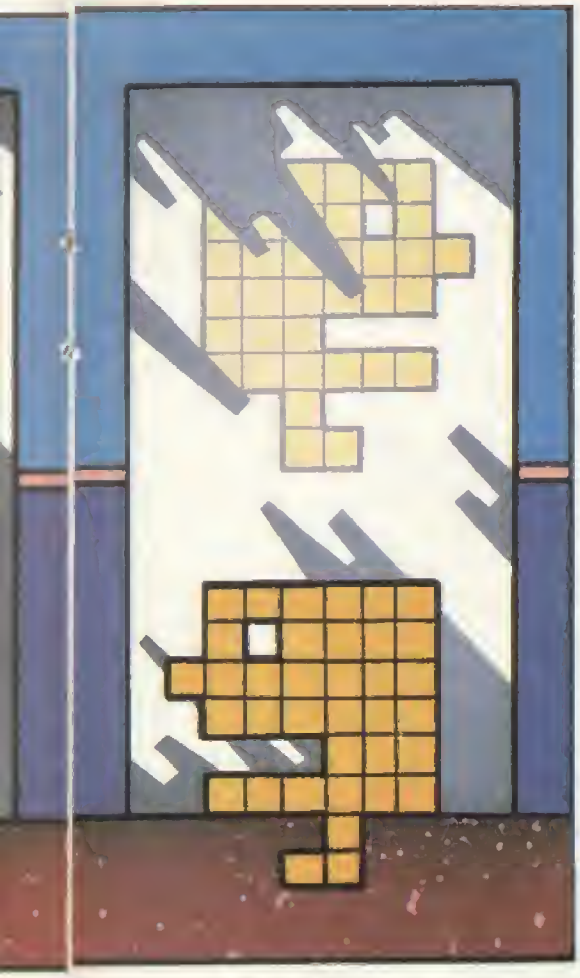
Para usar a impressora, pressione a tecla Z, seguida de D ou S, conforme queira valores de bytes ou uma cópia da tela, respectivamente. Caso outra tecla seja utilizada, o programa voltará à edição.

Se você não tiver uma impressora, não copie as linhas de 2570 a 2590. Acrescente, porém:

```
2570 GOTO 2000
```

Aqui está o resto do programa:

```
  5 CLEAR 31999
 12 LET T=0: FOR N=32000 TO
 32227: READ A: POKE N,A: LET
 T=T+A: NEXT N: IF T<>21691
 THEN  PRINT  FLASH 1;"ERRO NA
 S LINHAS 'DATA'": STOP
2020 PRINT AT 10,21;CHR$ 139;CH
R$ 131;CHR$ 135;AT 11,21;CHR$ 1
38;AT 11,23;CHR$ 133;AT 12,21;C
HR$ 142;CHR$ 140;CHR$ 141
2030 RAND USR 32000
2530 IF INKEY$="I" THEN  RAND U
SR 32092
2540 IF INKEY$="C" THEN  POKE 3
2106,0: RAND USR 32092: POKE 32
106,12
2550 IF INKEY$="M" THEN  RAND U
SR 32145
2560 IF INKEY$="R" THEN  RAND U
SR 32183
2570 IF INKEY$<>"Z" THEN  GOTO
2000
2575 INPUT "C(OPIA DA TELA) OU
B(YTES)? "; LINE Z$
```



```
2580 IF Z$<>"C" AND Z$<>"B" THE
N  GOTO 2000
2590 IF Z$="C" THEN  COPY : GOT
O 2000
2600 LET CH=65: FOR N=USR "A" T
O USR "U"+7 STEP 8
2610 LET TA=0: LPRINT CHR$ CH:
FOR M=N TO N+7: LPRINT TAB TA;P
EEK M;: LET TA=TA+4: NEXT M
2620 LPRINT : LET CH=CH+1: NEXT
 N
9100 DATA 62,2,205,1,22,62,22,2
15,62,8,215,175,215,33,11,72,22
1,33,118,72
9110 DATA 6,8,197,6,8,14,128,17
5,50,91,125,126,254,1,40,7,58,9
1,125,129
9120 DATA 50,91,125,203,57,35,1
6,239,58,91,125,221,119,0,229,2
21,229,62,23,215
9130 DATA 62,5,215,33,90,125,20
5,40,26,62,13,215,221,225,225,1
7,24,0,25,221
9140 DATA 229,209,20,213,221,22
5,193,16,189,201,0,0,33,11,72,6
,8,197,6,8
9150 DATA 197,126,254,1,229,40,
12,6,7,62,1,119,36,16,252,54,25
5,24,13,6
9160 DATA 4,54,85,36,54,171,36,
16,248,37,54,255,225,193,35,16,
219,17,24,0
9170 DATA 25,193,16,209,201,33,
11,72,6,8,197,6,4,17,7,0,197,22
9,126,25
9180 DATA 78,119,225,113,35,27,
27,193,16,242,17,28,0,25,193,16
,229,205,92,125
9190 DATA 195,92,125,221,33,11,
74,33,235,72,6,8,197,6,8,197,22
1,126,0,119
9200 DATA 221,35,6,32,43,16,253
,193,16,241,17,24,0,221,25,17,1
,1,25,193
9210 DATA 16,226,205,92,125,195
,92,125
```

Os valores nas linhas **DATA** são programas em linguagem de máquina colocados no alto da memória com **POKE**. Ali existem três rotinas diferentes para rodar, espelhar e colocar o caractere em tamanho real na tela, junto com o valor dos bytes.

Tome cuidado na digitação desses valores. Qualquer erro pode ser fatal. Por isso mesmo, o programa verifica as linhas **DATA**, provocando uma interrupção caso detecte algum erro.

USO DOS BLOCOS EM OUTROS PROGRAMAS

O programa do artigo anterior já incluía funções de gravação e leitura em fita. Por meio delas, podemos dispor de uma versão permanente dos blocos criados. O uso correto deste programa permite a criação de muitos bancos cheios de novos caracteres.

O Spectrum é capaz de utilizar 21 UDG de cada vez. Se precisar de um número maior, você terá duas opções: criar vários bancos de memória, ou redefinir o conjunto de blocos a todo instante. Explicaremos mais tarde como isso é feito, mas, agora, adiantaremos alguns detalhes.

Desde que saiba como modificar o "apontador de UDG", você pode utilizar quantos caracteres quiser. O apontador é uma variável interna do Spectrum que diz ao computador onde encontrar os bytes correspondentes aos blocos definidos pelo usuário.

Mesmo que tenha gravado os bytes criados pelo programa a partir de um determinado endereço, você poderá trazer este conjunto de caracteres de volta da fita para qualquer outro endereço. Assim, se quiser usar três diferentes bancos de caracteres ao mesmo tempo, bastará carregá-los em posições diferentes da memória.

Cada banco criado pelo programa tem 168 bytes de comprimento. Por isso, é necessário carregar cada conjunto de caracteres em posições que tenham uma distância de pelo menos 168 bytes entre si; caso contrário, um banco poderá apagar parte de outro. Para carregar um conjunto de caracteres gravado pelo programa, digite:

```
LOAD "" CODE (endereço inicial)
```

Independentemente do endereço a partir do qual o banco tenha sido gravado, pode-se colocá-lo em qualquer parte da RAM.

Para usar os bancos que carregou, modifique o valor do apontador de UDG. Em artigo posterior, trataremos do assunto em detalhes.

É importante adiantar, ainda, que ao usarmos mais de um banco de caracteres em um programa deveremos proteger, por meio de um comando **CLEAR**, a área de memória ocupada. Isso também será explicado mais tarde.

Feitas as modificações, o programa terá várias novas funções. Você notará a primeira delas quando for guardar um bloco recém-criado no banco: o programa imprimirá, ao ladó do quadriculado, o valor do byte correspondente ao padrão de cada linha do bloco, acompanhado do valor do byte de cor e do código da cor de frente e de fundo daquela linha.

O programa completo permite ainda que você limpe o quadriculado por meio da tecla A, evitando o trabalho de apagar ponto por ponto.

Outra alternativa interessante que o programa oferece é a de se obter, com a tecla E, a imagem ao espelho do bloco que está no quadriculado. Essa função facilita bastante a produção de desenhos simétricos.

Muito útil também é a possibilidade de rodar o bloco 90 graus, o que nos possibilita criar um torpedo, por exemplo, e apontá-lo em todas as direções, usando a tecla V.

Com a tecla I, pode-se inverter todo o padrão do bloco. Pressionando-a, os pontos acesos se apagam e os outros se acendem. A tecla Y tem o mesmo efeito, mas troca a cor de frente com a cor de fundo, não atuando, assim, no estado dos pontos. Em outras palavras, o I modifica apenas os bytes do padrão do bloco, enquanto o Y modifica os bytes de cor — e esta é uma diferença muito importante, embora não se reflita no efeito obtido. A última função utiliza uma impressora para reproduzir o padrão do bloco que está no quadriculado, bem como os valores dos bytes de padrão e cor. Para imprimir, pressiona-se a tecla L.

```
40 GOSUB 6000
135 IF K$="V" THEN GOSUB 4000:G
OTO 200
145 IF K$="A" THEN GOSUB 4500:G
OTO 200
155 IF K$="Y" THEN GOSUB 5000:G
OTO 200
165 IF K$="L" THEN GOSUB 5500:G
OTO 200
180 IF K$="I" THEN GOSUB 3000:G
OTO 200
190 IF K$="E" THEN GOSUB 3500:G
OTO 200
1550 GOSUB 9000:FOR I=0 TO 7:P(
I)=0:FOR J=0 TO 7
1595 II=64+I*8:PRESET(12,II):PR
INT#1,P(I)
1605 PRESET(56,II):PRINT#1,C(I)
:PRESET(176,II):PRINT#1,(C(I)-(
C(I)AND15))/16:PRESET(220,II):P
RINT#1,C(I)AND15
1610 NEXT I
1615 II=II+10:PRESET(12,II):PRI
NT#1,"BYTE"
1620 PRESET(56,II):PRINT#1,"COR
"
1630 PRESET(164,II):PRINT#1,"FR
ENTE"
1640 PRESET(216,II):PRINT#1,"FU
NDO"
1650 PRESET(8,160):PRINT#1,"Qua
lquer tecla para continuar"
1660 X$=INKEY$:IF X$="" THEN 16
60
1670 LINE(0,64)-(88,160),15,BF
1680 LINE(164,64)-(255,160),15,
BF
1690 GOSUB 9100:RETURN
3000 FOR I=0 TO 7:FOR J=0 TO 7
3010 B(I,J)=B(I,J)XOR1
3020 GOTO 1080
3500 FOR I=0 TO 7:FOR J=0 TO 7
3510 T(I,J)=B(I,7-J)
3520 NEXT J,I:FOR I=0 TO 7:FOR
J=0 TO 7
3530 B(I,J)=T(I,J):GOTO 1080
4000 FOR I=0 TO 7:FOR J=0 TO 7
4010 T(I,J)=B(J,7-I)
4020 GOTO 3520
4500 FOR I=0 TO 7:FOR J=0 TO 7
4510 B(I,J)=0
4520 GOTO 1080
5000 FOR I=0 TO 7
5010 Y1=C(I)AND15
5020 Y2=(C(I)-Y1)/16
5030 C(I)=Y1*16+Y2
5040 NEXT I:FOR I=0 TO 7:FOR J=
0 TO 7
5050 GOTO 1080
5500 LPRINT TAB(10);"BYTE";
5510 LPRINT TAB(15);" COR";
5520 LPRINT TAB(20);"FRENTE";
5530 LPRINT TAB(28);"FUNDO"
5550 FOR I=0 TO 7:P(I)=0:FOR J=
0 TO 7
5560 P(I)=P(I)+B(I,J)*2^(7-J)
5570 IF B(I,J)=1 THEN LPRINT "X
"; ELSE LPRINT ".";
5580 NEXT J
5590 LPRINT TAB(10);P(I);
5600 LPRINT TAB(16);C(I);
5610 LPRINT TAB(22);(C(I)-(C(I)
AND15))/16;
5620 LPRINT TAB(28);C(I)AND15
5630 NEXT I
5640 LPRINT:LPRINT:LPRINT
5650 RETURN
6000 FOR I=0 TO 7
6010 C(I)=C*16+F:NEXT:
6020 RETURN
```

Não há função automática que permita o aproveitamento do banco de blocos gravado em fita. Para isso, precisamos da ajuda de algumas linhas escritas em BASIC.

A elaboração de um programa desse tipo exige que se conheça bem a organização da memória de vídeo do MSX. Em artigo futuro trataremos deste assunto em detalhes.

ALGUMAS INFORMAÇÕES

Para os usuários mais avançados, adiantamos que um banco gravado pelo programa pode ser lido a qualquer momento. Inicialmente, deve-se proteger o alto da memória com:

```
CLEAR 200, &HD000
```

Em seguida, carrega-se o banco digitando:

```
BLOAD "CAS:"
```

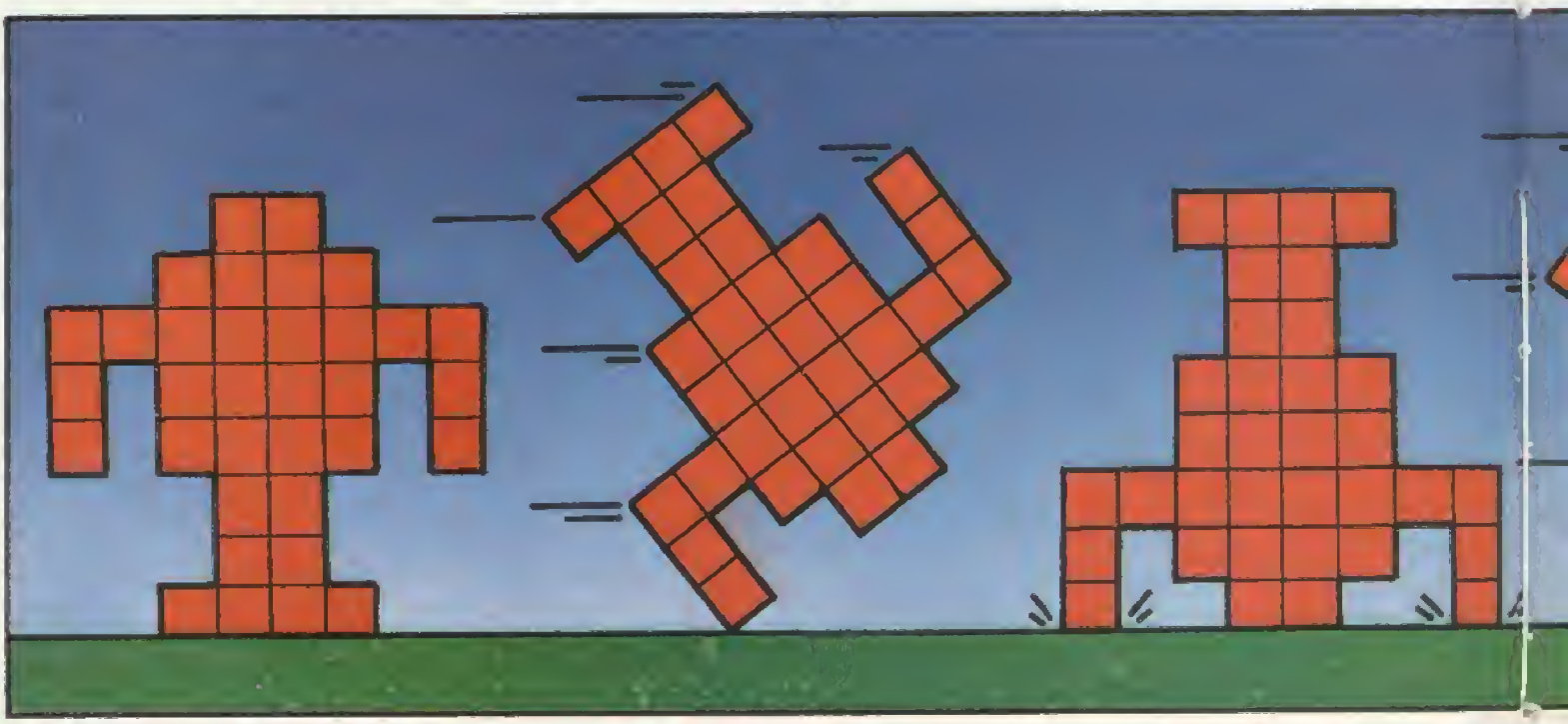
Com isso, o banco volta à mesma posição de memória onde foi criado. Os oito bytes correspondentes ao padrão do bloco guardado na posição **X** podem, então, ser encontrados a partir do seguinte endereço:

```
&HD100 + X*8
```

E os oito bytes de cor do bloco guardado na posição **X** do banco se encontram a partir de:

```
&HE100 + X*8
```

O passo seguinte consiste em transferir, com a ajuda de **VPOKE**, os bytes de padrão para a tabela de padrões — **BASE(12)** — e os bytes de cor para a ta-



bela de cores — **BASE(11)**. Maiores esclarecimentos serão dados num próximo artigo.

O programa conta agora com várias outras funções de edição. Uma delas é a de limpeza do quadriculado, disponível pela tecla C. Já não será preciso, portanto, apagar ponto por ponto.

A tecla U tem função semelhante, mas apaga apenas o oitavo bit, que controla o grupo de cores.

Duas outras novas funções contribuem para diminuir um pouco as dificuldades. Pressionando a tecla J, todo o padrão se desloca para a esquerda; a letra K tem efeito contrário. Utilizando essas teclas, pontos que estavam em colunas ímpares passam a ocupar colunas pares e vice-versa. Não havendo dois pontos adjacentes, o procedimento inverterá a cor dos pontos.

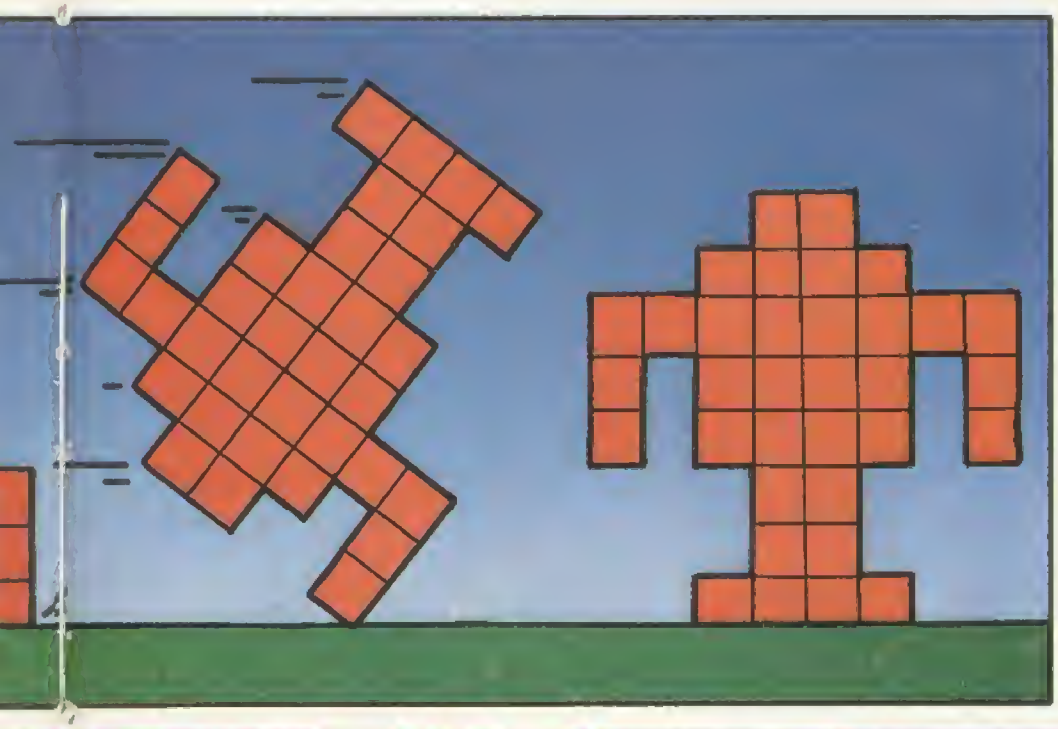
Para criar figuras simétricas, temos a tecla E, que produz a imagem ao espelho do bloco que está no quadriculado. Ao usá-la, tenha sempre em mente as regras de definição de cores. Para rodar o padrão do quadriculado 90 graus, pressione a tecla V. Essas funções devem ser combinadas com outras, uma vez que modificam o oitavo bit, que não participa do padrão, mas da determinação de cores.

Dispomos ainda de duas funções de inversão: apagar o que está aceso e vice-versa. A tecla I inverte todo o bloco, enquanto a tecla O se restringe ao oitavo bit. O programa também exige os valores dos bytes, para quem quiser usá-los em linhas **DATA**. A tecla D mostra uma linha com os dados de um bloco, no rodapé da tela. Tendo uma impressora, os usuários do Apple podem obter uma cópia usando W.

```
40  DIM T(8,8)
135  IF K$ = "V" THEN  GOSUB 40
00: GOTO 50
145  IF K$ = "U" THEN LL = 7: G
OSUB 2500: GOTO 50
155  IF K$ = "J" THEN  GOSUB 45
00: GOTO 50
160  IF K$ = "I" THEN LL = 0: G
OSUB 3000: GOTO 50
165  IF K$ = "K" THEN  GOSUB 50
00: GOTO 50
170  IF K$ = "C" THEN LL = 0: G
OSUB 2500: GOTO 50
175  IF K$ = "D" THEN  GOSUB 60
00: GOTO 50
180  IF K$ = "O" THEN LL = 7: G
OSUB 3000: GOTO 50
185  IF K$ = "W" THEN  PR# 1: G
OTO 6000: PR# 0: GOTO 50
190  IF K$ = "E" THEN  GOSUB 35
00: GOTO 50
2500  FOR I = 0 TO 7: FOR J = L
L TO 7
2510 B(I,J) = 0: GOTO 1020
3000  FOR I = 0 TO 7
3005  FOR J = LL TO 7
3010  IF B(I,J) = 0 THEN B(I,J)
 = 1: GOTO 1020
3020 B(I,J) = 0: GOTO 1020
3500  FOR I = 0 TO 7: FOR J = 0
 TO 6
3510 T(I,J) = B(I,6 - J): NEXT
J,I
3520  FOR I = 0 TO 7: FOR J = 0
 TO 6
3530 B(I,J) = T(I,J): GOTO 1020
4000  FOR I = 0 TO 7: FOR J = 0
 TO 7
4010 T(I,J) = B(J,7 - I): NEXT
J,I
4020  FOR I = 0 TO 7: FOR J = 0
 TO 7
4030 B(I,J) = T(I,J): GOTO 1020
4500  FOR I = 0 TO 7: FOR J = 0
 TO 6
4510 B(I,J) = B(I,J + 1): GOTO
1020
5000  FOR I = 0 TO 7: FOR J = 0
 TO 6
5010 T(I,7 - J) = B(I,6 - J): N
EXT J,I
5020  FOR I = 0 TO 7: FOR J = 0
 TO 6
5030 B(I,J) = T(I,J): GOTO 1020
6000  HOME : VTAB 22: INPUT "QU
AL O NUMERO DO BLOCO ?";N: IF N
 > 320 THEN 1000
6020  HOME
6030  VTAB 21: PRINT "DATA ";
6040  FOR I = 0 TO 7: PRINT  PE
EK (E + N * 8 + I);: IF J <  >
7 THEN  PRINT " ,";
6050  NEXT : VTAB 23: GET X$
6060  HOME : RETURN
```

USO DOS BLOCOS EM OUTROS PROGRAMAS

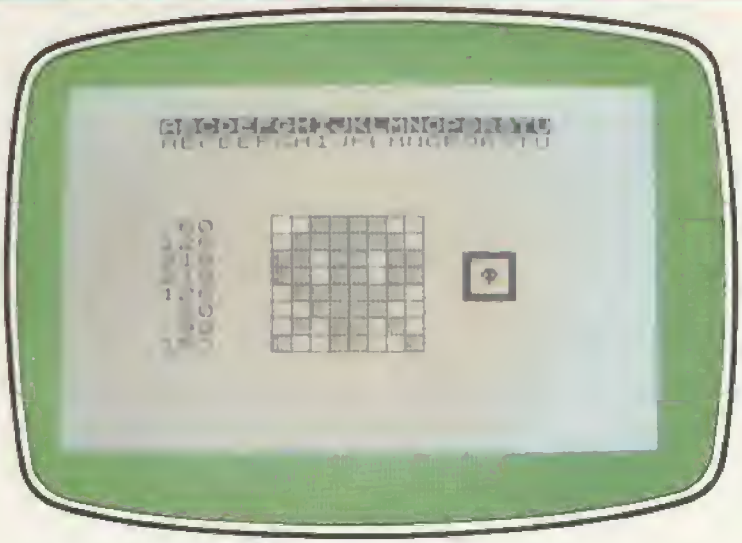
Além dos problemas na exibição de cores, o Apple tem uma organização de memória de vídeo um tanto complicada. Deixaremos, assim, para um artigo futuro as explicações sobre o uso dos blocos em outros programas.



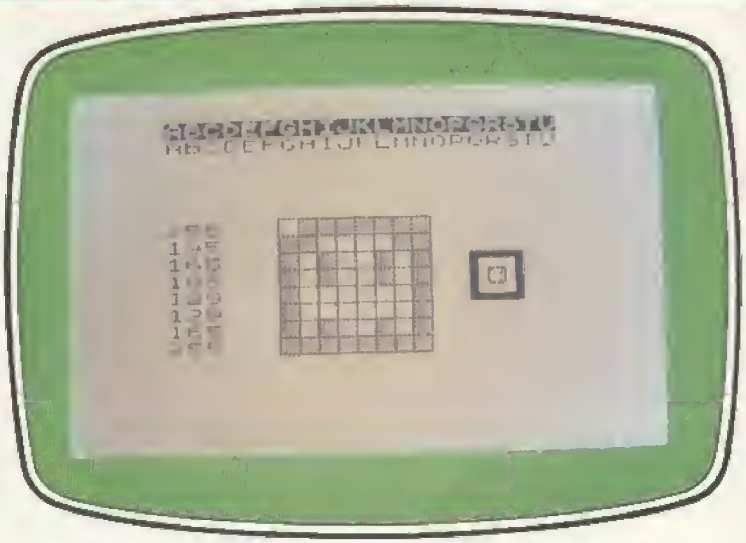
No artigo anterior vimos a primeira metade de um programa cujo objetivo é facilitar a criação de blocos gráficos. Apresentamos aqui a outra metade, que acrescenta ao programa novos recursos de edição

Inicialmente, carregue o programa antigo e adicione as linhas:

```
30 T=0:FOR K=0 TO 14:READ N:T=T
+N:POKE K+31100,N:NEXT:READC:IF
 T<>C THEN END
40 T=0: FOR K=0 TO 84:READ N:T=
T+N:POKE 31150+K,N:NEXT:READ C:
IF T<>C THEN END
70 DATA 141,72,142,123,12,236,1
29,237,193,140,123,84,38,247,57
,1974
80 DATA 141,22,142,123,84,51,67
,198,3,166,130,167,194,90,38,24
9,51,70,140,123,12,38,240,57,18
9,179,237,52
90 DATA 6,31,3,142,123,14,134,3
,183,121,68,230,192,134,8,74,88
```



As novas rotinas acrescentam várias funções ao programa gerador de gráficos; entre elas, a rotação de figuras.



Com o programa completo, você poderá também inverter seus gráficos, utilizando a cor do fundo da tela.

```
,73,125,121,23,39,4,88,73,128,4
,68
100 DATA 102,132,125,121,23,39,
3,68,102,132,77,38,230,48,31,12
2,121,68,38,219,48,6,140,123,86
,38,207,53,192,8285
170 PRINT"JOYSTICK OU TECLADO (
J OU T)?";
180 A$=INKEY$:IF A$<>"J" AND A$
<>"T" THEN 180
190 IF A$="J" THEN JY=1
350 IF JY=1 THEN GOSUB 1000 ELS
E GOSUB 1500
1000 PUT(X1,Y1)-(X1+5*T-1,Y1+4)
,C1,NOT
1010 IF (PEEK(65280)AND 1)=0 GO
SUB 2000
1020 IF JOYSTK(1)=0 THEN Y=Y-1
1030 IF JOYSTK(1)=63 THEN Y=Y+1
1040 IF JOYSTK(0)=0 THEN X=X-T
1050 IF JOYSTK(0)=63 THEN X=X+T
1060 RETURN
2100 GET(216,70)-(239,93),A
2110 GOSUB 3000:GOTO 2070
2200 A$=INKEY$:IF A$<>"S" AND A
$<>"P" THEN 2200
2210 CLS:DN=0:IF A$="P" THEN DN
=-2
2220 FOR K=0 TO 14:FOR R=0 TO 2
2230 PRINT #DN,PEEK(VARPTR(A(0)
)+K*3+R);:NEXT:PRINT#DN:NEXT
2240 IF DN<0 THEN 2260
2250 A$=INKEY$:IF A$="" THEN 22
50
2260 FOR K=15 TO 23: FOR P=0 TO
 2
2270 PRINT #DN,PEEK(VARPTR(A(0)
)+K*3+R);:NEXT:PRINT #DN:NEXT
2280 A$=INKEY$:IF A$="" AND DN=
0 THEN 2280
2290 SCREEN 1,ST:RETURN
2800 POKE 30999,T-1:N=USR2(VARP
TR(A(0)))
2810 GOSUB 3000:GOTO 2070
2900 POKE 30999,T-1:N=USR1(VARP
TR(A(0)))
2910 GOSUB 3000:GOTO 2070
```

Complete a linha 10 com:

```
:DEFUSR1=31100:DEFUSR2=31150
```

Se executarmos o programa agora, poderemos utilizar as novas funções. Mas é conveniente, antes de qualquer coisa, verificar se há algum erro em linhas **DATA**.

Três funções foram somadas ao repertório do programa: espelhar, inverter e rodar o bloco gráfico.

Para obter a imagem do bloco ao espelho, basta pressionar a tecla M. O caractere é invertido da esquerda para a direita. Essa função é muito útil quando utilizamos dois blocos justapostos para produzir uma figura maior e simétrica. Basta criar um dos caracteres, guardá-lo no banco e, em seguida, obter a imagem ao espelho.

A função de rotação é similar. Apertando R, rodamos o bloco 180 graus. Podemos, assim, mover a figura para cima e para baixo, o que nos permite usar o gerador para criar uma versão do bloco "de cabeça para baixo".

A inversão da cor de todos os pontos constitui outra nova opção oferecida pelo programa. Em **PMODE4**, o efeito é fácil de prever: o preto se torna verde (ou cinza) e vice-versa. Em **PMODE3**, azul e amarelo são invertidos (o que é azul fica amarelo e vice-versa). O mesmo ocorre com vermelho e verde, cinza e laranja, ciano e magenta.

As inversões de cor provocam alterações surpreendentes no aspecto de certos blocos. Uma vez que nos acostumemos com as regras de mudança de cor, a função será muito útil.

É possível, também, executar uma mudança de cor durante a edição, pressionando-se a tecla V. A alteração, no caso, resulta de uma inversão no tipo de tela — **SCREEN** — que estiver sendo utilizada.

Outro novo recurso do programa, a impressão dos valores dos bytes dos caracteres guardados no banco, mostra-se especialmente útil quando se pretende colocar tais valores em linhas **DATA**. A tecla P ativa a função; os valores podem ser exibidos na tela ou, então, copiados por uma impressora. Após teclarmos P, o programa espera que pressionemos P ou S. P utiliza a impressora e S, a tela.

USO DOS BLOCOS EM OUTROS PROGRAMAS

Para utilizar os blocos gravados em fita dentro de um outro programa, carregue o conteúdo da fita e, em seguida, guarde os padrões em matrizes com **CLOADM** e **GET**.

Infelizmente, você precisará repetir o procedimento sempre que quiser usar o programa com os blocos gráficos anteriormente gravados.

Para evitar o trabalho de ler os blocos da fita todas as vezes, só há uma alternativa: transferir os números correspondentes aos blocos para linhas **DATA** e incorporá-las ao programa em questão.

Assim, embora não pareça, pode ser mais conveniente recorrer à função de impressão para digitar linhas **DATA**. Os números ali contidos deverão ser colocados pelo programa, usando **POKE**, na tela, e transferidos para matrizes por comandos **GET**. Depois disso, os blocos estarão disponíveis por meio de comandos **PUT**.

O BASIC NA MEMÓRIA

- COMO É ARMAZENADO UM PROGRAMA EM BASIC?
- O USO DO PEEK
- O QUE SIGNIFICAM OS CÓDIGOS

Como o computador armazena um programa em BASIC na memória? Satisfaça sua curiosidade: com um programa bem simples, você poderá listar os códigos secretos usados por seu micro.

Já tivemos a oportunidade de examinar como se organiza o espaço total de memória dos microcomputadores (veja artigo da página 174). Em geral este espaço divide-se internamente em uma série de áreas, cada qual com uma função específica.

Uma dessas áreas da memória RAM é reservada para os programas em BASIC do usuário. Eles começam sempre em um mesmo endereço absoluto, cujo valor varia conforme a linha do micro. Assim, quando o computador recebe um comando **RUN**, **LIST** ou **LLIST**, por exemplo, já "sabe" onde o programa se inicia, podendo começar a listá-lo ou interpretá-lo. Em muitos computadores, esse endereço de início está contido em um par de apontadores da RAM, em um lugar prefixado. Alterando-se seus valores com alguns **POKE**, pode-se mudar o local da memória onde o programa em BASIC está armazenado.

A listagem dos códigos internos que seu microcomputador utiliza para armazenar esses programas constitui um ótimo exercício para quem está começando a aprender programação em linguagem de máquina.

O PROGRAMA

Neste artigo, apresentaremos um programa bem simples, baseado no comando **PEEK**, que nos permite examinar os códigos numéricos decimais contidos nas locações das memórias ROM ou RAM, e seu equivalente em ASCII. Você poderá, assim, identificar as partes dos programas que contêm códigos diferentes do ASCII.

Os computadores de algumas linhas já dispõem de programas-monitores residentes, que servem perfeitamente para este propósito (é o caso do TRS-80, do Apple II e do TK-2000). Por isso, os programas seguintes destinam-se apenas aos micros das linhas ZX-81, Spectrum, TRS-Color e MSX, que não dispõem desse recurso.

TK

```
10 FOR I=7681 TO 7851 STEP 5
20 PRINT I;TAB(6);":";
30 FOR J=I TO I+4:PRINT USING
"### ";PEEK(J);:NEXT J
60 PRINT TAB(25);
70 FOR J=I TO I+4
80 IF PEEK(J)>31 AND PEEK(J)<
129 PRINT CHR$(PEEK(J)); ELSE
PRINT " ";
120 NEXT J:PRINT:NEXT I
```

O programa acima funciona no TRS-Color. Para o MSX, modifique a linha 10 para:

```
10 FOR I=-32765 TO -32595 STEP
   5
```

S

```
10 FOR i=23755 TO 23925 STEP
5
20 PRINT i;TAB 6;":";
30 FOR j=I TO i+4:PRINT PEEK
j;:NEXT j
60 PRINT TAB(25);
70 FOR j=i TO i+4
80 IF PEEK j>31 AND PEEK j<96
THEN GOTO 110
90 PRINT CHR$ PEEK(j);:GOTO 120
110 PRINT " ";
120 NEXT j
130 PRINT
140 NEXT i
```

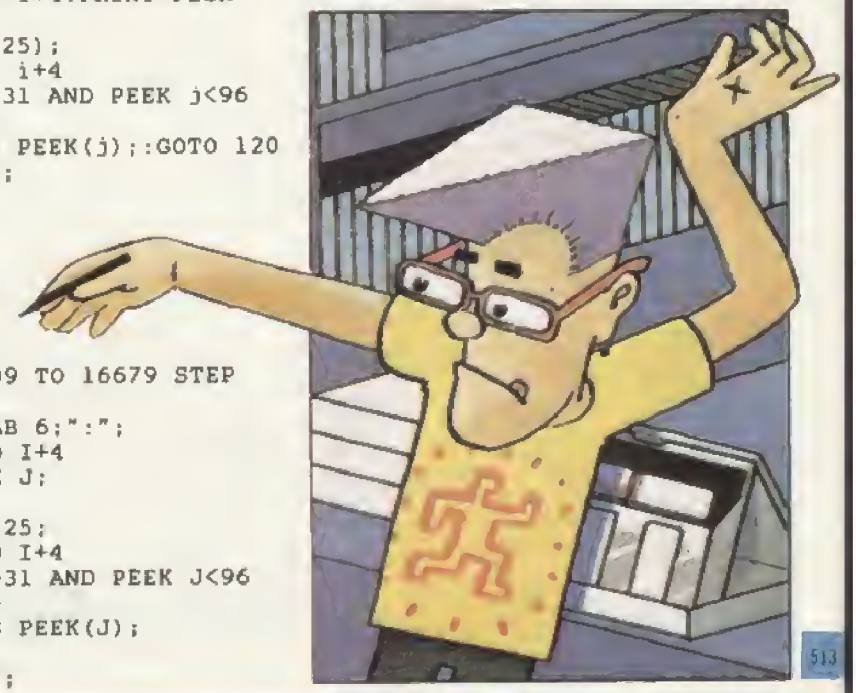
S

```
10 FOR I=16509 TO 16679 STEP
5
20 PRINT I;TAB 6;":";
30 FOR J=I TO I+4
40 PRINT PEEK J;
50 NEXT J
60 PRINT TAB 25;
70 FOR J=I TO I+4
80 IF PEEK J>31 AND PEEK J<96
THEN GOTO 110
90 PRINT CHR$ PEEK(J);
100 GOTO 120
110 PRINT " ";
120 NEXT J
130 PRINT
140 NEXT I
```

O laço **FOR...NEXT** que começa na linha 10 coloca o endereço absoluto da localização da memória a ser listada, a partir do ponto inicial de todo programa BASIC, em incrementos de 5. A linha 20 imprime o endereço absoluto da primeira memória de um grupo de 5, enquanto os laços das linhas 30 e 70 se encarregam, respectivamente, da impressão do conteúdo numérico e dos caracteres ASCII correspondentes.

Observe que as palavras-chave do BASIC não são armazenadas caractere por caractere, mas sim como códigos numéricos de 1 byte — os chamados *tokens*, em inglês. Cada código corresponde a uma das palavras reservadas do BASIC, existentes no manual do micro. Eles são traduzidos de volta para as palavras-chave quando o programa é listado com **LIST** ou **LLIST**. Armazenam-se em ASCII, por sua vez, as constantes, nomes de variáveis, caracteres de pontuação, símbolos matemáticos e tudo o que estiver entre aspas.

Como exercício, procure identificar os códigos das palavras reservadas no programa listado.



O DIVERTIDO JOGO DA COBRA

Ajude uma cobra faminta a encontrar alimento. Enquanto vai abocanhando os números que o computador coloca na tela, o desnutrido animal se transforma, aos poucos, numa gigantesca serpente.

O jogo da cobra consiste num tipo clássico de videogame, de regras simples mas muito interessante e cheio de surpresas. Além disso, sua programação dispensa o uso de linguagem de máquina — o jogo desafia o tempo, sendo um dos mais facilmente programáveis em BASIC.

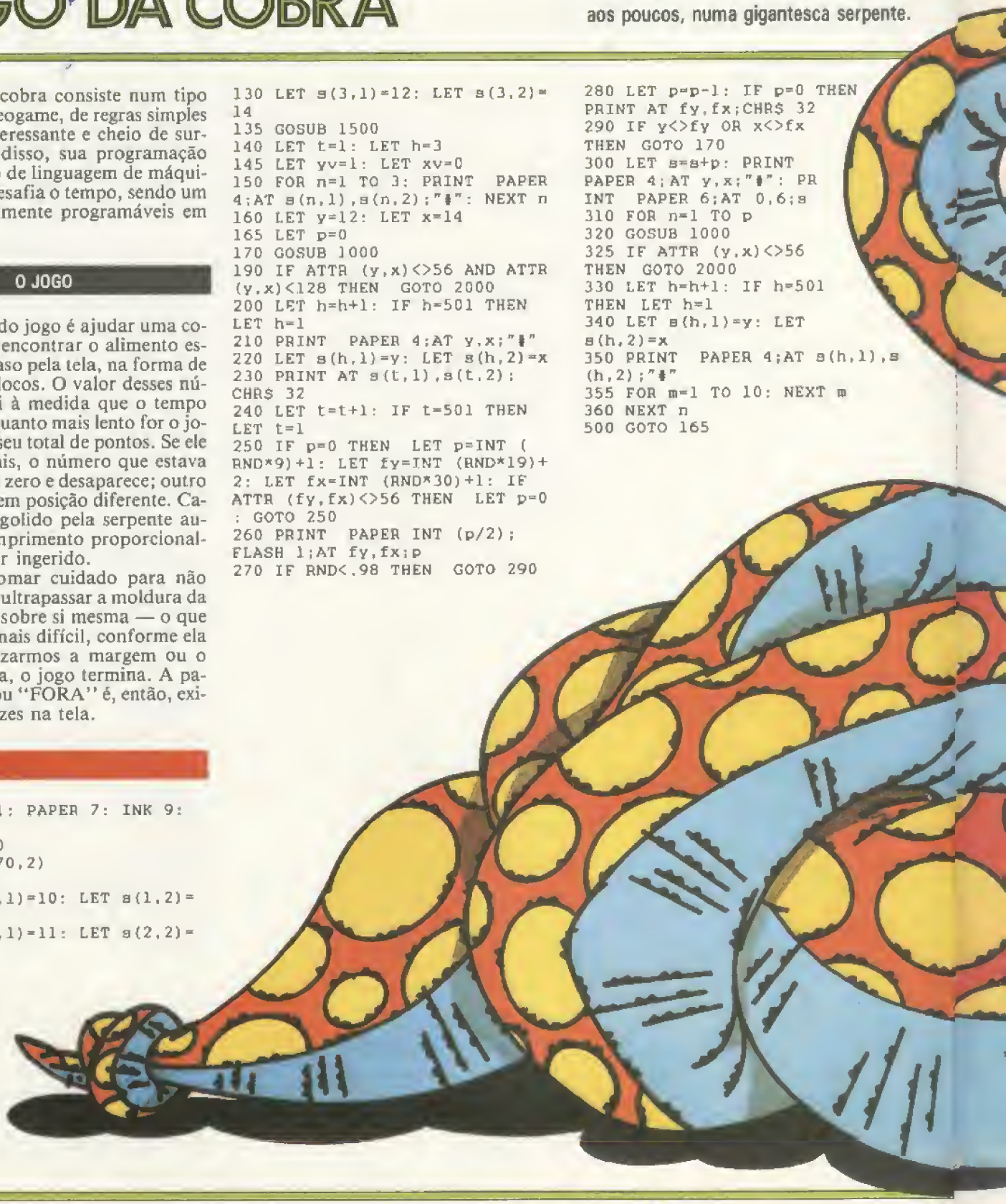
O JOGO

O objetivo do jogo é ajudar uma cobra faminta a encontrar o alimento espalhado ao acaso pela tela, na forma de números ou blocos. O valor desses números diminui à medida que o tempo passa; assim, quanto mais lento for o jogador, menor seu total de pontos. Se ele demorar demais, o número que estava na tela chega a zero e desaparece; outro número surge em posição diferente. Cada número engolido pela serpente aumenta seu comprimento proporcionalmente ao valor ingerido.

Devemos tomar cuidado para não deixar a cobra ultrapassar a moldura da tela ou passar sobre si mesma — o que fica cada vez mais difícil, conforme ela cresce. Se cruzarmos a margem ou o corpo da cobra, o jogo termina. A palavra "FIM" ou "FORA" é, então, exibida várias vezes na tela.

S

```
 10 BORDER 1: PAPER 7: INK 9:
 CLS
 20 LET hs=0
 30 DIM s(570,2)
100 LET s=0
110 LET s(1,1)=10: LET s(1,2)=
14
120 LET s(2,1)=11: LET s(2,2)=
14
130 LET s(3,1)=12: LET s(3,2)=
14
135 GOSUB 1500
140 LET t=1: LET h=3
145 LET yv=1: LET xv=0
150 FOR n=1 TO 3: PRINT  PAPER
4;AT s(n,1),s(n,2);"#": NEXT n
160 LET y=12: LET x=14
165 LET p=0
170 GOSUB 1000
190 IF ATTR (y,x)<>56 AND ATTR
(y,x)<128 THEN  GOTO 2000
200 LET h=h+1: IF h=501 THEN
LET h=1
210 PRINT  PAPER 4;AT y,x;"#"
220 LET s(h,1)=y: LET s(h,2)=x
230 PRINT AT s(t,1),s(t,2);
CHR$ 32
240 LET t=t+1: IF t=501 THEN
LET t=1
250 IF p=0 THEN  LET p=INT (
RND*9)+1: LET fy=INT (RND*19)+
2: LET fx=INT (RND*30)+1: IF
ATTR (fy,fx)<>56 THEN  LET p=0
: GOTO 250
260 PRINT  PAPER INT (p/2);
FLASH 1;AT fy,fx;p
270 IF RND<.98 THEN  GOTO 290
280 LET p=p-1: IF p=0 THEN
PRINT AT fy,fx;CHR$ 32
290 IF y<>fy OR x<>fx
THEN  GOTO 170
300 LET s=s+p: PRINT
PAPER 4;AT y,x;"#": PR
INT  PAPER 6;AT 0,6;s
310 FOR n=1 TO p
320 GOSUB 1000
325 IF ATTR (y,x)<>56
THEN  GOTO 2000
330 LET h=h+1: IF h=501
THEN  LET h=1
340 LET s(h,1)=y: LET
s(h,2)=x
350 PRINT  PAPER 4;AT s(h,1),s
(h,2);"#"
355 FOR m=1 TO 10: NEXT m
360 NEXT n
500 GOTO 165
```



- UMA VERSÃO BASIC DO JOGO DA COBRA
- ESPALHE OS NÚMEROS E FAÇA A COBRA ABOCANHÁ-LOS
- COMO A FIGURA CRESCE

```
1000 LET a$=INKEY$
1010 IF a$="q" THEN  LET yv=-1:
 LET xv=0
1020 IF a$="a" THEN  LET yv=1:
LET xv=0
1030 IF a$="o" THEN  LET xv=-1:
 LET yv=0
1040 IF a$="p" THEN  LET xv=1:
LET yv=0
1050 LET y=y+yv: LET x=x+xv: RE
      TURN
      1500 FOR n=22560 TO 225
      91: POKE n,16: POKE n+6
      40,16: NEXT n
      1510 FOR n=22592 TO 225
      92+32*19  STEP 32: POKE
       n,16: POKE n+31,16: NE
      XT n
1520 PRINT  PAPER 1;AT 0,0;"SCO
RE "; PAPER 6;s; PAPER 1;TAB 14
;"RECORDE  "; PAPER 6;hs; PAPER
 1;TAB 31;" "
1590 RETURN
2000 PRINT AT 0,0;: FOR n=1 TO
88: PRINT  FLASH 1; PAPER 2; IN
K 6;"FORA"; PAPER 6; INK 2;"FOR
A";: SOUND .005,60-n: NEXT n
2010 IF s>hs THEN  LET hs=s
2020 CLS : PRINT AT 8,10;"SCORE
: ";s;AT 11,8;"RECORDE: ";hs
2030 PRINT  INVERSE 1;AT 16,2;"
Qualquer tecla para recomecar"
2040 PAUSE 0: CLS : GOTO 100
```

A linha 10 cuida das cores da moldura, do fundo e dos caracteres. Também limpa a tela. As linhas 20 e 100 zeram o recorde e o score.

A linha 30 **DIM**ensiona a matriz **S**, usada para armazenar as coordenadas da cobra. Inicialmente, esta ficará nas coordenadas (10,14), (11,14), (12,14), colocada nos primeiros elementos da matriz pelas linhas 110 e 130.

A linha 135 chama uma sub-rotina que prepara a tela — linha 1500. Para o desenho da moldura, os valores são colocados na tabela de atributos com o auxílio de **POKE**. A linha 1500 traça duas linhas na tela: uma em cima e outra embaixo; a 1510 desenha as linhas laterais. A linha 1520 escreve o recorde e o score.

A linha 140 determina onde estão a cabeça e o rabo da cobra — **h** e **t**, respectivamente — dentro da matriz **s**. Essas variáveis indicam ao computador os elementos da matriz em que as coordenadas da cabeça e do final do corpo da

cobra podem ser encontradas. Elas são chamadas de "apontadores" (*pointers*). Conforme a cobra se move, o computador reajusta os dois apontadores e coloca as novas coordenadas no elemento adequado da matriz.

MOVIMENTO E DIREÇÃO

As variáveis **xv** e **yv** indicam a direção em que está indo a cobra. Elas podem assumir três valores diferentes: 0, quando a cobra não está indo naquela direção; 1, quando a cobra se dirige para a direita ou para baixo; e −1, quando a cobra está seguindo para a esquerda ou para cima. Pode-se ver, desse modo, que a linha 145 faz com que a serpente siga para cima no começo do jogo.

A linha 150 desenha a cobra, inicialmente, com apenas três caracteres. A posição atual da cabeça é dada por **x** e **y**, variáveis também usadas para detectar colisões. A linha 160 posiciona a cabeça em (14,12). A linha 165 faz com que o valor do número a ser mostrado — **p** — seja zero.

A linha 170 chama a sub-rotina que movimenta o cursor — linha 1000. Q e A movimentam a serpente na vertical, enquanto O e P o fazem na horizontal. Pressionando essas teclas, alteramos os valores de **xv** e **yv**; a linha 1050 modifica a posição da cabeça adicionando **xv** e **yv** a **x** e **y**.

Quando termina a sub-rotina, o programa retorna para a linha 190, que usa o comando **ATTR** para verificar se a cabeça está sendo colocada em uma posição que não é de cor branca — a cor do fundo — ou não é cintilante — como são os números. Se essas duas condições não forem satisfeitas, a cobra deve ter cruzado a moldura ou seu próprio corpo. Neste caso, o programa vai para a linha 2000, início da sub-rotina que informa o final do jogo.

Se a nova posição da cabeça for válida, o programa prossegue na linha 200, que incrementa o apontador da cabeça. Se o apontador indica um elemento da matriz maior do que sua dimensão, o apontador passa a valer 1. A linha 210 desenha a cabeça em sua nova posição e a linha 220 coloca as coordenadas desta posição na matriz **s**, no elemento indicado pelo apontador da cabeça — **h**. O rabo da cobra é mantido no tamanho correto por meio da impressão de espaços em branco que apagam os caracteres das posições já ocupadas. Isso é feito pela linha 230, que obtém as coordenadas corretas na matriz **s**. A linha 240, por sua vez, aumenta o valor do apontador da cauda — **t** — verificando se ele ultrapassou a dimensão da matriz.

Se não houver um número na tela para a cobra comer, a linha 250 escolhe um — **p** —, com posição e valor aleatórios. Se as coordenadas escolhidas não correspondem a uma área em branco da tela — **ATTR** <> 56 —, o valor e a posição de **p** são escolhidos novamente. A linha 260 escreve o número na tela cintilando, isto é, com atributo **FLASH**.

À medida que o tempo passa, o valor do número que está na tela vai diminuindo. A linha 270, que compara um número aleatório com 0.98, introduz um certo elemento de acaso na velocidade com que ocorre a diminuição. Na maioria dos casos, a linha que diminui o valor do número — 280 — é pulada. Essa linha verifica se o valor de **p** ainda não chegou a zero, logo após subtrair-lhe 1 — em caso afirmativo, um espaço em branco é colocado no lugar do número na tela. Se a cabeça da cobra não está na mesma posição que o número, completa-se o laço e o programa volta à linha 170.

Se a serpente engolir o número, o programa segue na linha 300, que soma o número engolido ao score. Este é colocado na tela, assim como a cabeça.

O laço **FOR...NEXT** das linhas 310 a 360 adiciona ao corpo da cobra uma quantidade de caracteres igual ao valor ingerido. A cada volta do laço, a linha 320 chama a rotina de movimentação do cursor; a linha 325 verifica se a cabeça ocupa uma posição legal; a linha 330 incrementa o valor do apontador da cabeça; a linha 340 coloca a nova posição na matriz **s**, e a linha 350 imprime a cabeça. Nenhum caractere é apagado, como aconteceria normalmente, servindo todos para acrescentar segmentos ao corpo da cobra. Como estes também nunca são apagados, a cobra teria movimentos bem mais rápidos, se não houvesse o atraso provocado pela linha 355.

A porção final do programa — que começa na linha 2000 — é a sub-rotina que cuida do encerramento do jogo. A linha 2000 escreve vários "FORA" na tela, ao mesmo tempo que emite uma série de sons. A linha 2010 atualiza o valor do recorde, se este foi batido, e a linha 2020 informa o placar. As linhas 2030 e 2040 permitem que o jogador inicie uma nova partida.

```
5 SCREEN 1:KEY OFF
10 M=512:DIMB(M)
15 RR=RND(-TIME)
20 L=3:GOSUB 600
30 GOTO 500
100 K=STICK(0):IF K=0 THEN K=L
110 NP=B(H)+32*(K=1)-32*(K=5)-(
K=3)+(K=7)
120 TE=VPEEK(NP)
130 IF TE=255 OR TE=HC OR TE=BC
 OR NP<BASE(5)+32 THEN VPOKE NP
,BC:E=1
140 L=K:FOR I=1 TO 50:NEXT:RETU
RN
200 R=INT(RND(1)*9)+1:RX=INT(RN
D(1)*13)+2:RY=INT(RND(1)*29)+2:
RP=RX*32+RY
210 IF VPEEK(BASE(5)+RP)<>32 TH
EN 200
220 VPOKE BASE(5)+RP,48+R:P=1:R
ETURN
250 R=R-1:IF R=0 THEN VPOKE BAS
E(5)+RP,32:P=0:RETURN ELSE VPOK
E BASE(5)+RP,48+R:RETURN
300 SC=SC+R:LOCATE 22,0:PRINT S
C;:P=0
310 SL=SL+1:VPOKE NP,BC
320 H=H-1:IF H=0 THEN H=M
330 B(H)=NP
340 FOR J=1 TO R
350 PLAY"L64CEF"
360 GOSUB 100:IF E=1 THEN 800
370 H=H-1:IF H=0 THEN H=M
380 B(H)=NP:VPOKE NP,BC
390 NEXT
400 VPOKE NP,HC:RETURN
500 IF P=0 THEN GOSUB 200 ELSE
IF INT(RND(1)*150)+1<SL THEN GO
SUB 250
510 GOSUB 100:IF E=1 THEN 800
520 IF TE=48+R THEN VPOKE B(H),
BC:GOSUB 300
530 VPOKE B(H),BC:VPOKE B(T),32
540 H=H-1:IF H=0 THEN H=M
550 T=T-1:IF T=0 THEN T=M
560 B(H)=NP:VPOKE NP,HC
570 GOTO 500
600 BG=INT(RND(1)*13)+1:FG=INT(
RND(1)*13+1):P=0:SC=0:SL=1:E=0
610 IF BG=FG THEN 600
620 COLOR 15,BG,BG:CLS
630 VPOKE BASE(6)+4,240+BG
640 VPOKE BASE(6)+31,FG*16+FG
650 FOR J=0 TO 31:PLAY"T255L640
4AG":VPOKE BASE(5)+J,255:VPOKE
BASE(5)+J+736,255:NEXT:FOR J=32
 TO 736 STEP 32:PLAY"02DA":VPOK
E BASE(5)+J,255:VPOKE BASE(5)+3
1+J,255:NEXT
660 LOCATE 3,0:PRINT"RECORDE=";
HS;:PRINT TAB(15);"SCORE = 0 ";
670 HC=2:BC=215:T=3:H=1
680 VPOKE BASE(6),FG*16+BG
690 VPOKE BASE(6)+26,FG*16+BG
700 B(1)= BASE(5)+239:VPOKE B(1
),HC
710 B(2)=B(1)+32:VPOKE B(2),BC
720 B(3)=B(2)+32:VPOKE B(3),BC
730 RETURN
800 COLOR 15,INT(RND(1)*14)+1:C
LS:PLAY"O1ACDEFGACDEFG":FOR K=1
 TO 408:PRINT "FIM ";:NEXT:PLAY
"O4ABCEDFGO3ABCDEFGO2ABCDEFG"
810 IF HS<SC THEN HS=SC
820 COLOR 1,15,15:CLS:LOCATE 9,
8:PRINT"SCORE =";SC:LOCATE 8,12
:PRINT"RECORDE =";HS
830 A$=INKEY$:LOCATE 3,20:PRINT
"<RETURN> para continuar"
```

```
840 A$=INKEY$:IF A$<>CHR$(13) T
HEN 840 ELSE 20
```

Ao contrário de outros jogos vistos em INPUT, para os quais utilizamos a tela gráfica, optamos aqui pela tela de textos de 32 colunas, já que a cobra é feita de caracteres.

A linha 5 seleciona a tela e desliga o menu das teclas de funções da parte inferior da tela. A linha 10 dimensiona a matriz **B** de maneira que esta acomode no máximo 512 caracteres — o maior tamanho que a cobra poderá ter. Note que os elementos dessa matriz não correspondem às posições de caracteres na tela. Cada um deles contém as coordenadas de uma parte da cobra. A linha 15 "dá a partida" no gerador de números randômicos; a 20 manda o programa para a linha 600, que prepara as condições iniciais do jogo.

Na linha 600, **BG** é a cor de fundo da tela e **FG**, a cor da moldura, bem como da cobra. **P=0** significa que nenhum número está sendo exibido; **SC=0**, que o score é zero; **SL=0** quer dizer que o nível de dificuldade é zero e **E=0** informa ao computador que a cobra permanece dentro dos limites da tela, não tendo cruzado nem a moldura, nem seu próprio corpo. Todas essas variáveis são utilizadas ao longo do programa para que certas sub-rotinas sejam chamadas no momento adequado.

A linha 610 verifica se a cor de fundo é igual à da moldura — em caso afirmativo, duas novas cores são escolhidas. A linha 620 seleciona as cores da tela — caracteres brancos, fundo e bordas de acordo com **BG**. Além disso, limpa a tela.

Na linha 630, um comando **VPOKE** assegura que o caractere de espaço em branco — código 32 — tenha cor de frente branca e cor de fundo igual a **BG**. Na linha 640, a mesma técnica é empregada para fazer com que o caractere 255, usado para desenhar a moldura, tenha cor de frente e de fundo igual a **FG**. A linha 650 desenha a moldura, enquanto emite alguns sons. A linha 660 mostra o placar.

Na linha 670, **HC** é o código do caractere da cabeça da cobra e **BC**, o código do caractere usado no corpo da mesma. As variáveis **H** e **T** indicam os elementos de **B()** que contêm as coordenadas da cabeça e da ponta da cauda da serpente. São chamadas de "apontadores" (*pointers*).

Ainda usando o comando **VPOKE** para modificar os valores da tabela de cores da tela de 32 colunas, as linhas 680 e 690 fazem com que os caracteres da cabeça e do corpo da cobra tenham cor de frente igual a **FG** e cor de fundo igual a **BG**. As linhas 700 a 720 colocam a cabeça e dois segmentos do corpo na tela, de modo que a cobra tenha apenas estas partes a cada início de partida.

ROTINA PRINCIPAL

Da sub-rotina o programa retorna à linha 30, que o envia para a rotina principal, na linha 500. Essa linha verifica, inicialmente, se na tela há um número para a cobra comer. **P** é o indicador de número na tela; se seu valor for zero, a sub-rotina da linha 200 é chamada. Se houver um número na tela, um valor qualquer entre 1 e 150 é comparado com o nível de dificuldade **SL**. Se o número gerado for menor que **SL**, o programa vai à sub-rotina 250.

A sub-rotina que começa na linha 200 coloca um número qualquer na tela, em uma posição escolhida ao acaso. A linha 200 seleciona o número — R — entre 1 e 9, bem como suas coordenadas **RX** e **RY**. Esses dois valores são usados para calcular **RP**, posição do número na tabela de nomes da tela 1. Antes que o número seja impresso na tela, a linha 210 verifica se na sua posição há algum outro caractere além do de espaço — código 32. Dessa maneira, evita que o número seja colocado sobre a moldura ou sobre o corpo da serpente. A linha 220 escreve o número na tela, usando **VPOKE** para colocar o código correspondente ao número na tabela de nomes — **BASE (5)** — da tela 1. Consultando uma tabela ASCII, você verá que os caracteres de 0 a 9 têm códigos de 48 a 57.

A sub-rotina que começa na linha 250 faz o valor do número na tela ir diminuindo até que o jogador leve a cobra até ele. Se o número chegar a zero antes da cobra comê-lo, um espaço em branco será colocado em seu lugar. Enquanto isso não acontecer, valores sucessivamente mais baixos substituem os anteriores, na mesma posição. A sub-rotina retorna para a linha 510.

Essa linha chama outra sub-rotina, na linha 100, que recebe os comandos de movimentação do cursor. Podemos usar tanto as teclas com setas como um joystick, desde que o comando **STICK** da linha 100 seja modificado (veja o artigo da página 348). O **IF...THEN** verifica se a cobra cruzou a moldura ou seu próprio corpo. Em caso afirmativo, o programa vai para a linha 800, que cuida do fim da partida.

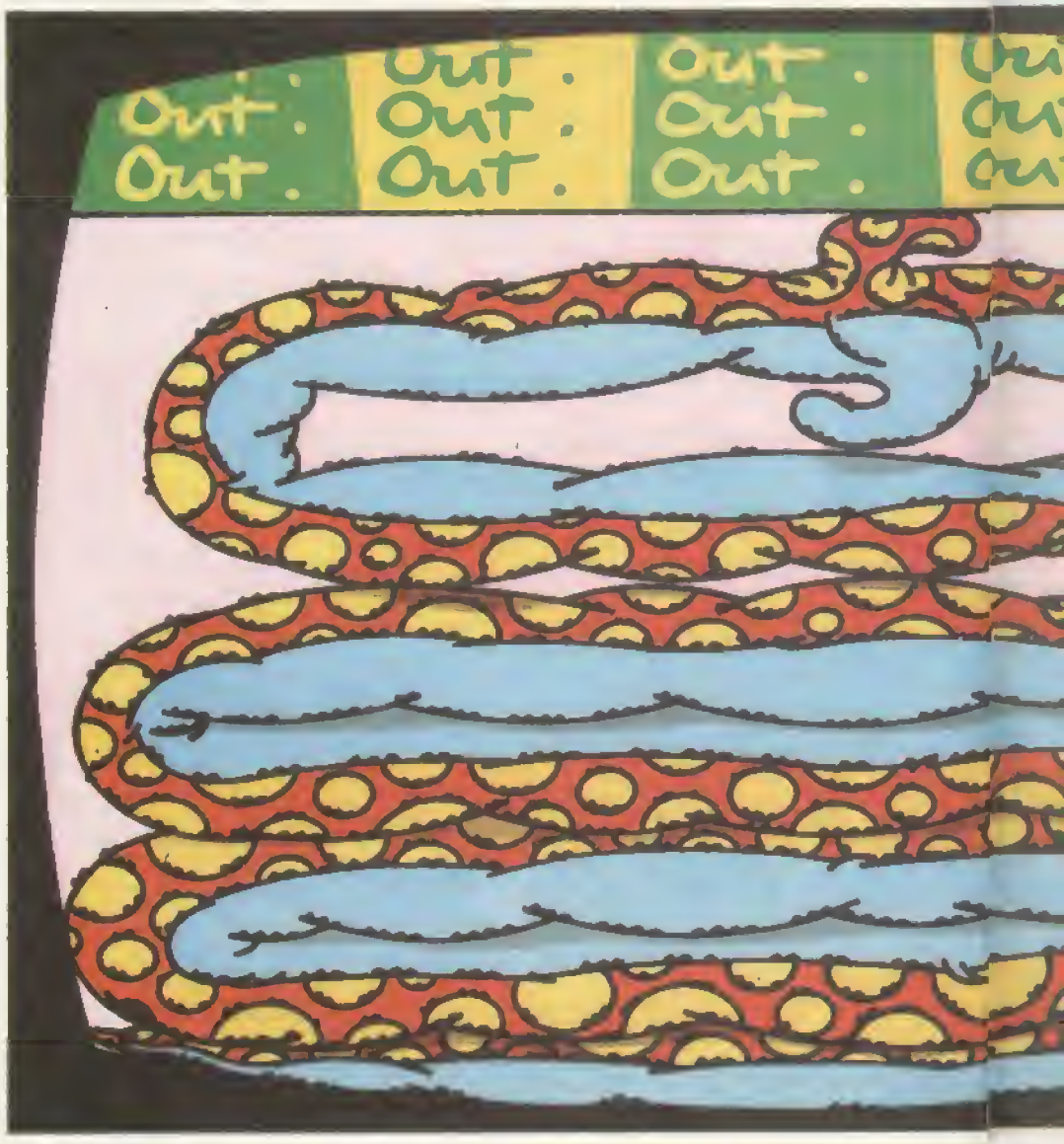
A linha 520 testa se o número foi engolido pela cobra, somando 48 a **R** e comparando o resultado com **TE**. Somamos 48 a **R** para obter o código do caractere correspondente ao número que estava na tela. **TE**, obtido com **VPEEK**, corresponde ao código do caractere que estava na posição que a cabeça da cobra ocupa agora. Caso o número tenha sido realmente engolido, um caractere do corpo é colocado em sua posição. O programa vai, então, para a sub-rotina

da linha 300, que calcula o novo score, aumenta o tamanho da cobra proporcionalmente ao número engolido — laço entre as linhas 340 e 390 — e coloca outro número na tela.

As linhas 540 e 550 modificam os apontadores da cabeça e da ponta da cauda, ajustando seus valores, caso tenham se reduzido a zero. A linha 560 coloca o caractere da cabeça na tela.

A rotina que cuida do encerramento da partida começa na linha 800, que limpa a tela, usando uma cor ao acaso, e produz um som enquanto a palavra "FIM" é impressa várias vezes. No fim da operação, outro som é produzido. A linha 810 atualiza o recorde, antes que as linhas 820 a 840 ofereçam ao jogador a oportunidade de jogar novamente.

```
5  HOME : GR
10 M = 500: DIM B(M)
20  GOSUB 600
30  GOTO 500
100  GET K$: IF K$ <  > "P" AND
 K$ <  > "L" AND K$ <  > "X" AN
D K$ <  > "Z" THEN K$ = L$
110 NP = B(H) - 40 * (K$ = "P")
 + 40 * (K$ = "L") - (K$ = "Z")
 + (K$ = "X")
120 TE =  SCRN( NP -  INT (NP /
 40) * 40, INT (NP / 40))
130  IF TE = 9 OR TE = 11 OR TE
 = 8 OR NP < 40 THEN  COLOR= 11
: PLOT NP -  INT (NP / 40) * 40
, INT (NP / 40):E = 1
140 L$ = K$: RETURN
200 R =  INT ( RND (1) * 9) + 1
:RX =  INT ( RND (1) * 37) + 2:
RY =  INT ( RND (1) * 37) + 2
210  IF  SCRN( RX,RY) <  > 0 TH
EN 200
220  VTAB 22: HTAB 31: PRINT "B
ONUS ";R:P = 1
230  COLOR= 6: PLOT RX,RY: RETU
RN
250 R = R - 1: IF R = 0 THEN  C
OLOR= 0: PLOT RX,RY:P = 0: RETU
RN
260  HTAB 31: VTAB 22: PRINT "B
ONUS ";R: RETURN
300 SC = SC + R: HTAB 7: VTAB 2
2: PRINT SC:P = 0
310 SL = SL + 1: COLOR= 8: PLOT
 NP -  INT (NP / 40) * 40, INT
(NP / 40)
320 H = H - 1: IF H = 0 THEN H
= M
330 B(H) = NP
340  FOR J = 1 TO R
350 XX =  PEEK ( - 16336)
360  GOSUB 100: IF E = 1 THEN 8
00
370 H = H - 1: IF H = 0 THEN H
= M
380 B(H) = NP: COLOR= 8: PLOT N
P -  INT (NP / 40) * 40, INT (N
P / 40)
390  NEXT
400  COLOR= 11: PLOT NP -  INT
(NP / 40) * 40, INT (NP / 40)
410  RETURN
500  IF P = 0 THEN  GOSUB 200:
GOTO 510
505  IF  INT ( RND (1) * 150) +
 1 < SL THEN  GOSUB 250
510  GOSUB 100: IF E = 1 THEN 8
00
520  IF TE = 6 THEN  COLOR= 0:
PLOT B(H) -  INT (B(H) / 40) *
40, INT (B(H) / 40): GOSUB 300
530  COLOR= 8: PLOT B(H) -  INT
 (B(H) / 40) * 40, INT (B(H) /
40): COLOR= 0: PLOT B(T) -  INT
 (B(T) / 40) * 40, INT (B(T) /
40)
540 H = H - 1: IF H = 0 THEN H
= M
550 T = T - 1: IF T = 0 THEN T
= M
560 B(H) = NP: COLOR= 11: PLOT
NP -  INT (NP / 40) * 40, INT (
NP / 40)
570  GOTO 500
600 L$ = "X":P = 0:SC = 0:SL =
1:E = 0
650  COLOR= 9: HLIN 0,39 AT 0:
VLIN 0,39 AT 0: VLIN 0,39 AT 39
: HLIN 0,39 AT 39
660  HOME : VTAB 22: PRINT "SCO
RE ";SC; TAB( 15);"RECORDE ";HS
670 T = 3:H = 1: COLOR= 11
680 B(1) = 300: PLOT B(1) -  IN
T (B(1) / 40) * 40, INT (B(1) /
 40)
690  COLOR= 8
700 B(2) = 340: PLOT B(2) -  IN
T (B(2) / 40) * 40, INT (B(2) /
 40)
710 B(3) = 380: PLOT B(3) -  IN
T (B(3) / 40) * 40, INT (B(3) /
 40)
720  RETURN
800  HGR : TEXT : HOME : FOR I
= 1 TO 500:XX =  PEEK ( - 16336
```



```
): PRINT "FIM  ";: NEXT
805  IF HS < SC THEN HS = SC
810  HOME : HTAB 15: VTAB 8: PR
INT "SCORE = ";SC: HTAB 14: VTA
B 12: PRINT "RECORDE = ";HS
820  HTAB 6: VTAB 20: PRINT "<R
ETURN> PARA JOGAR NOVAMENTE": G
ET A$
830  IF A$ =  CHR$ (13) THEN  H
OME : GR : GOTO 20
880  GOTO 820
```

Para o jogo no Apple e no TK-2000, escolhemos a tela de baixa resolução, já que a cobra é feita de blocos.

A linha 5 seleciona a tela e limpa sua parte inferior. A 10 dimensiona a matriz B de modo a acomodar no máximo quinhentos caracteres — o maior tamanho que a cobra poderá ter. Note que os elementos da matriz B não correspondem às posições dos blocos na tela. Cada um deles contém as coordenadas de uma parte da cobra. A linha 20 manda o programa para a linha 600, que prepara as condições iniciais do jogo.

Na linha 600, **P=0** significa que nenhum bloco está sendo mostrado; **SC=0**, que o score é zero; **SL=0** quer dizer que o nível de dificuldade é zero e **E=0** informa ao computador que a cobra permanece dentro dos limites da tela, não tendo cruzado nem a moldura, nem seu próprio corpo. Todas essas variáveis são utilizadas ao longo do programa para que certas sub-rotinas sejam chamadas no momento adequado.

A linha 650 desenha a moldura. A linha 660 mostra o placar.

Na linha 670, 11 é a cor da cabeça da cobra. **H** e **T** são variáveis usadas para indicar os elementos de **B ( )** que contêm as coordenadas da cabeça e da ponta da cauda da serpente. Essas duas variáveis são chamadas de "apontadores" (*pointers*).

As linhas 680 a 710 colocam a cabeça e dois segmentos do corpo na tela, de modo que a cobra tenha apenas estas partes a cada início de partida. A cor do corpo da cobra é 8.

BLOCOS NA TELA

Da sub-rotina o programa retorna à linha 30, que o envia para a rotina principal, na linha 500. Essa linha verifica, inicialmente, se há comida (bloco) para a cobra na tela. **P** é o indicador de bloco na tela; se seu valor for zero, a sub-rotina da linha 200 é chamada. Se houver um bloco na tela, um número qualquer entre 1 e 150 é comparado com o nível de dificuldade **SL**. Se o número gerado for menor que **SL**, a linha 505 manda o programa para a sub-rotina 250.

A sub-rotina que começa na linha 200 coloca um bloco na tela, em uma posição escolhida ao acaso. A linha 200 seleciona o número — R — entre 1 e 9, bem como duas coordenadas **RX** e **RY**. A linha 210 verifica se a posição tem a cor 0, preta, indicando que não há nada no local. A linha 220 imprime na porção inferior da tela o valor do bloco que ali está (bônus). A linha 230 coloca um bloco de cor 6 em um ponto qualquer da tela.

A sub-rotina da linha 250 diminui gradativamente o valor do bônus, até que o jogador consiga fazer a cobra chegar à comida. Se o número for reduzido a zero antes que a cobra coma o bloco, um espaço em branco é colocado em seu lugar. Enquanto isso não ocorre, valores mais baixos são sucessivamente exibidos. A sub-rotina retorna para a linha 510.

Esta linha chama outra sub-rotina, na linha 100, que recebe os comandos de movimentação do cursor. Usamos as teclas P, L, Z e X como de costume. O **IF...THEN** verifica se a cobra cruzou a moldura ou seu próprio corpo; em caso afirmativo, o programa vai para a linha 800, que cuida do encerramento da partida.

A linha 520 testa se a comida foi engolida pela cobra, comparando a cor do bloco com **TE**. Usamos **SCRN** na linha 120 para obter **TE**, cor do bloco que estava na posição que a cabeça da cobra ocupa. Caso a comida tenha sido realmente engolida, um bloco do corpo é colocado em sua posição. O programa vai, então, para a sub-rotina da linha 300, que calcula o novo score, aumenta o tamanho da cobra proporcionalmente ao bônus — laço entre as linhas 340 e 390 — e, por fim, coloca outro bloco na tela.

As linhas 540 e 550 mudam os apontadores da cabeça e da ponta da cauda, ajustando seus valores, caso tenham se reduzido a zero. A linha 560 coloca o bloco da cabeça da cobra na tela.

A rotina que cuida do final da partida começa na linha 800, que limpa a tela e provoca um ruído (no Apple), enquanto a palavra "FIM" é impressa várias vezes. A linha 805 atualiza o recorde, antes que as linhas 810 a 830 ofereçam ao jogador a oportunidade de jogar novamente.

T

```
10 M=512:DIM B(M)
20 GOSUB 600
30 GOTO 500
100 K$=INKEY$:IF K$="" THEN K$=
L$
110 K=ASC(K$):NP=B(H)-32*(K=10)
```

```
+32*(K=94)-(K=9)+(K=8)
120 TE=PEEK(NP)
130 IF TE=FG OR TE=HC OR TE=BC
OR NP<1056 THEN POKE NP,HC:E=1
140 LS=K$:RETURN
200 R=RND(9):RX=RND(13)+1:RY=RN
D(29)+1:RP=RX*32+RY
210 IF PEEK(1024+RP)<>BG THEN 2
00
220 R$=RIGHT$(STR$(R),1):PRINT@
RP,R$;:P=1:RETURN
250 R=R-1:IF R=0 THEN PRINT @RP
,CHR$(BG);:P=0:RETURN ELSE PRIN
T @RP,RIGHT$(STR$(R),1);:RETURN
300 SC=SC+R:PRINT @26,SC;:P=0
310 SL=SL+1:POKE NP,BC
320 H=H-1:IF H=0 THEN H=M
330 B(H)=NP
340 FOR J=1 TO R
350 PLAY "L255CEF"
360 GOSUB 100:IF E=1 THEN 800
370 H=H-1:IF H=0 THEN H=M
380 B(H)=NP:POKE NP,BC
390 NEXT
400 POKE NP,HC:RETURN
500 IF P=0 THEN GOSUB 200 ELSE
IF RND(159)<SL THEN GOSUB 250
510 GOSUB 100: IF E=1 THEN 800
520 IF TE=R+112 THEN POKE B(H),
BC: GOSUB 300
530 POKE B(H),BC:POKE B(T),BG
540 H=H-1:IF H=0 THEN H=M
550 T=T-1:IF T=0 THEN T=M
560 B(H)=NP:POKE NP,HC
570 GOTO 500
600 BG=RND(7)+1:FG=RND(7)+1:LS=
CHR$(94):P=0:SC=0:SL=1:E=0
610 IF BG=FG THEN 600
620 CLS BG:BG=BG-1:FG=FG-1
630 BG=143+16*BG
640 FG=143+16*FG
650 FOR J=1024 TO 1055:PLAY "T2
55L255O4AG":POKE J,FG:POKE J+48
0,FG:NEXT:FOR J=1056 TO 1472 ST
EP 32:PLAY"O2DA":POKE J,FG:POKE
 J+31,FG:NEXT
660 PRINT @3,"RECORDE=";HS;:PRI
NT @15,"SCORE= 0 ";
670 HC=ASC("*")+64:BC=ASC("#")+
64:T=3:H=1
680 B(1)=1263:POKE B(1),HC
690 B(2)=1295:POKE B(2),BC
700 B(3)=1327:POKE B(3),BC
710 RETURN
800 CLS RND(8): PLAY"O1ACDEFGAC
DEFG":FOR K=1 TO 408:PRINT " FO
RA!! ";:NEXT:PLAY"O4ABCDEFGO3AB
CDEFGO2ABCDEFG"
810 IF HS<SC THEN HS=SC
820 CLS:PRINT @73,"SCORE    =";S
C:PRINT @230,"   RECORDE =";HS
830 A$=INKEY$:PRINT @450,"PRESS
IONE QUALQUER TECLA PARA   RECO
MECAR"
840 A$=INKEY$: IF A$="" THEN 84
0 ELSE 20
```

Usamos a tela de textos, e não a gráfica, para o jogo no TRS-Color, já que a cobra é feita de caracteres.

A linha 10 dimensiona a matriz **B** de modo que esta acomode no máximo 512 caracteres — o maior tamanho que a cobra poderá ter. Note que os elementos da matriz **B** não correspondem às posições de caracteres na tela. Cada elemento contém as coordenadas de uma parte da cobra. A linha 20 manda o programa à linha 600, que prepara as condições iniciais do jogo.

Na linha 600, **BG** é a cor de fundo da tela e **FG**, a cor da moldura. **L$ = CHR$(94)** faz com que a cobra se movimente para cima, até que uma tecla seja pressionada. **P = 0** significa que nenhum número está sendo mostrado; **SC = 0**, que o score é zero; **SL = 0** quer dizer que o nível de dificuldade é zero, e **E = 0** informa ao computador que a cobra permanece dentro dos limites da tela, não tendo cruzado nem a moldura, nem seu próprio corpo. Todas essas variáveis são utilizadas ao longo do programa para que certas sub-rotinas sejam chamadas no momento adequado.

A linha 610 verifica se a cor de fundo é igual à da moldura, escolhendo duas novas cores se o par inicial for idêntico. A linha 620 usa **CLS BG** para colorir a tela. Subtrai 1 de **BG** e **FG**, numa preparação para as contas das linhas 630 e 640. Embora pareçam complexas, essas contas simplesmente convertem **BG** e **FG** em valores que possam ser colocados na tela com **POKE**, a fim de produzir a cor desejada.

A linha 650 desenha a moldura, enquanto produz alguns sons. A linha 660 mostra o placar.

Na linha 670, **HC** é o código do caractere da cabeça da cobra e **BC**, o código do caractere usado no corpo da mesma. **H** e **T** são variáveis usadas para indicar os elementos de **B( )** que contêm as coordenadas da cabeça e da ponta da cauda da serpente. Essas duas variáveis são chamadas de "apontadores" (*pointers*).

As linhas 680 a 700 colocam na tela a cabeça e dois segmentos do corpo, de modo que a cobra tenha apenas estas partes a cada início de partida.

Da sub-rotina o programa retorna para a linha 30, que o envia para a rotina principal na linha 500. Essa linha verifica, inicialmente, se há um número na tela para a cobra comer. **P** é o indicador de número na tela. Se seu valor for igual a zero, a sub-rotina da linha 200 é chamada. Se houver um número na tela, um número qualquer entre 1 e 150 é comparado com o nível de dificuldade **SL**. Se o número gerado for menor que **SL**, o programa vai à sub-rotina 250.

A sub-rotina que começa na linha 200 coloca na tela um número qualquer, em uma posição escolhida ao acaso. A linha 200 seleciona o número — R — entre 1 e 9, bem como suas coordenadas **RX** e **RY**. Esses dois valores são usados para calcular **RP**, posição do número na tabela de nomes da tela 1. Antes que o número seja impresso na tela, a linha 210 verifica se na sua posição há outra cor além da cor de fundo, para evitar uma superposição com a moldura ou com o corpo da serpente. A linha 220 cabe escrever o número na tela, e colocar 1 em **P**.

A sub-rotina que começa na linha 250 diminui gradativamente o valor do número na tela, até que o jogador consiga fazer a cobra comê-lo. Se o número for reduzido a zero, um espaço em branco é colocado em seu lugar. Enquanto isso não ocorre, valores sucessivamente mais baixos são colocados na mesma posição. A sub-rotina retorna para a linha 510.

Essa linha chama outra sub-rotina, na linha 100, que recebe os comandos de movimentação do cursor. O **IF...THEN** verifica se a cobra cruzou a moldura ou seu próprio corpo; em caso afirmativo, o programa vai para a linha 800, que cuida do fim da partida.

A COBRA CRESCE

A linha 520 testa se a cobra engoliu o número, somando 112 a **R** e comparando o resultado com **TE**. Somamos 112 a **R** para obter o código do caractere correspondente ao número que estava na tela. **TE** é obtido com **PEEK**, e corresponde ao código do caractere que estava na posição que a cabeça da cobra ocupa agora. Caso o número tenha sido realmente engolido, um caractere do corpo é colocado em sua posição. O programa vai, então, para a sub-rotina da linha 300, que calcula o novo score, aumenta o tamanho da cobra proporcionalmente ao número engolido — laço entre as linhas 340 e 390 — e coloca outro número na tela.

As linhas 540 e 550 modificam os apontadores da cabeça e da ponta da cauda, ajustando seus valores, caso eles tenham se reduzido a zero. A linha 560 coloca o caractere da cabeça da cobra na tela.

A rotina que cuida do encerramento da partida começa na linha 800, que limpa a tela usando uma cor ao acaso e produz algum som enquanto a palavra "FIM" é impressa várias vezes. Ao final da operação, outro som é produzido. A linha 810 se incumbe de atualizar o recorde, antes que as linhas 820 a 840 ofereçam ao jogador a oportunidade de jogar novamente.

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Craft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Os blocos gráficos definidos pelo usuário podem assumir a forma que quisermos. A imaginação é o limite.

CÓDIGO DE MÁQUINA
Apagando as linhas **REM** e os espaços inúteis, você tornará seus programas mais rápidos e econômicos. Veja como fazê-lo.

PERIFÉRICOS
Você quer adquirir uma impressora? Para uma boa escolha, analise as características dos diversos modelos.

PROGRAMAÇÃO BASIC
Conhecemos várias rotinas de ordenação. Qual é a melhor para o seu caso?